BACHELORARBEIT

vorgelegt an der Hochschule für angewandte Wissenschaften Würzburg-Schweinfurt
an der Fakultät Kunststofftechnik und Vermessung
zum Abschluss eines Studiums im Studiengang
Vermessung und Geoinformatik

Thema:

Trennung von parkenden und am Verkehr teilnehmenden Fahrzeugen basierend auf einer automatischen Verkehrserfassung aus Luftbildern

Angefertigt am:	Deutsches Zentrum für Luft- und Raumfahrt (DLR) Institut für Methodik der Fernerkundung - Photogrammetrie und Bildanalyse (IMF-PBA)
Betreuer:	Dr. rer.nat. Dominik Rosenbaum (DLR, IMF-PBA) Prof. Dr.-Ing. Ansgar Brunn (FHWS)
Prüfer:	Dr.rer.nat. Dominik Rosenbaum (DLR, IMF-PBA) Prof. Dr.-Ing. Ansgar Brunn (FHWS)
Abgabetermin:	22. Januar 2018

Eingereicht von
Julia Knöttner
aus Eltmann/Dippach
Matrikelnummer: 6014005
Würzburg, den 22. Januar 2018

Erklärung zur Bachelorarbeit

Hiermit versichere ich, dass die vorgelegte Bachelorarbeit selbstständig verfasst und noch nicht anderweitig zu Prüfungszwecken vorgelegt wurde. Alle verwendeten Quellen und Hilfsmittel sind angegeben. Wörtliche und sinngemäße Zitate wurden als solche gekennzeichnet.

Würzburg, den

Unterschrift des/der Studierenden

Einwilligung zur Überprüfung einer Arbeit mit der Plagiatserkennungssoftware PlagScan

Name: Knöttner Vorname: Julia Matr.Nr. 6014005

Adresse: Roßstadter Weg 3, 97483 Eltmann/Dippach

E-Mail: julia.knoettner@student.fhws.de Studiengang: Vermessung und Geoinformatik

Titel der Arbeit: Trennung von parkenden und am Verkehr teilnehmenden Fahrzeugen basierend auf einer automatischen Verkehrserfassung aus Luftbildern
Betreuer: Dr.rer.nat. Dominik Rosenbaum (DLR, IMF-PBA), Prof. Dr. Ing. Ansgar Brunn (FHWS)

Auf Grund der Zielvereinbarung der FHWS mit dem Bayerischen Staatsministerium für Bildung und Kultus, Wissenschaft und Kunst vom 19. März 2014, Ziffer 2.3, hat die Hochschule entschieden, Studien- und Abschlussarbeiten künftig durch die Plagiatserkennungssoftware PlagScan elektronisch auf Plagiate hin zu überprüfen.

Die zu überprüfenden Arbeiten werden an den Dienst PlagScan übermittelt, dort auf Übereinstimmung mit externen Quellen untersucht und zum Zweck des Abgleichs mit zukünftig zu überprüfenden Studien- und Prüfungsarbeiten gespeichert. **Die befristete Speicherung Ihrer Arbeit in der Datenbank sowie die Weitergabe Ihrer persönlichen Daten im Rahmen der Plagiatsprüfung ist nur mit Ihrer Einwilligung zulässig.**

Mit einer Unterschrift erkläre ich meine Einwilligung, dass

- **die von mir vorgelegte und verfasste Arbeit zum Zweck der Überprüfung auf Plagiate hin an PlagScan übermittelt und vorübergehend (5 Jahre) in der von PlagScan geführten Datenbank gespeichert wird;**
- **meine persönlichen Daten (Vorname, Name, studentische E-Mail-Adresse) zusammen mit dem Text digital gespeichert und verwendet werden. Diese Daten sind nur meiner Prüferin oder meinem Prüfer/meinen Prüferinnen oder Prüfern zugänglich.**

Hinweis:

Diese Einwilligungserklärung ist freiwillig. Sie haben die Möglichkeit die Erklärung abzulehnen. Durch die Verweigerung der Einwilligung kann bei Entfernung der persönlichen Angaben und Wahrung der urheberrechtlichen Vorgaben die Plagiatsprüfung nicht verhindert werden. Die Einwilligung zur Speicherung und Verwendung der persönlichen Daten kann jederzeit durch Erklärung gegenüber der Fakultät mit Wirkung für die Zukunft widerrufen werden.

Ort Datum Unterschrift

Aufgabenstellung

Ziel der Bachelorarbeit ist die Entwicklung von Modellen und Methoden, mit denen es innerstädtisch möglich ist, parkende Fahrzeuge von am Verkehr teilnehmenden (stehenden) Fahrzeugen in den aus Luftbildern generierten Straßenverkehrsdaten zu trennen. Dazu sollen sowohl die Vektordaten der automatischen Detektions- und Trackingalgorithmen verwendet werden, als auch weitere Merkmale, die wiederum aus den originalen Luftbildern generiert werden können. Darüber hinaus stehen weitere Daten zur Verfügung, wie z.B. die OSM- oder Navteq-Straßendatenbank oder ATKIS Daten.

Da die aus Luftbildern generierten Verkehrsdaten nicht nur zur Verkehrslagedarstellung, sondern auch zur Verkehrssimulation genutzt werden sollen, ist eine Trennung zwischen parkenden und am Verkehr teilnehmenden Fahrzeugen unerlässlich. Vielmehr ist auch zur Anzeige der aktuellen Verkehrssituation eine saubere Verkehrslagedarstellung ohne die gleichzeitig miterfassten, parkenden Fahrzeuge wünschenswert. Darüber hinaus kann eine separate Erfassung von parkenden Fahrzeugen zur Messung der Parkraumbelegung herangezogen werden, was wiederum die Verkehrslenkung z.B. in der Anreisephase von Großereignissen erleichtert.
Ein Grundstock an Luftbilddaten ist dafür bereits vorhanden, wobei auch in der laufenden Bachelorarbeit noch Befliegungen durchgeführt werden können, wenn die entwickelte Methodik spezielle Anforderungen an die Luftbilder, bspw. hinsichtlich ihrer Auflösung stellt.

Die entwickelten Modelle und Methoden sollen dann später in das operationelle System zur Verkehrserfassung aus Luftbildsequenzen übernommen werden.

Zusammenfassung

In dieser Arbeit wird eine Methodik entwickelt, die Fahrzeuge in „parkende“ und „am Verkehr teilnehmende“ unterteilt. Dies geschieht auf Basis der automatischen Verkehrserfassung des DLR und unter Berücksichtigung der benachbarten Fahrzeuge. Bei der Verkehrserfassung wurden die Fahrzeuge detektiert und getrackt, sodass für jedes wiedergefundene Fahrzeug mindestens zwei Positionen vorliegen. Mithilfe dieser Daten können Verkehrsparameter, wie beispielsweise die Fahrgeschwindigkeit, Fahrtrichtung und die zurückgelegte Fahrtstrecke ermittelt werden. Um eine Beziehung zu den benachbarten Fahrzeugen aufbauen zu können, wird eine Delaunay Triangulation durchgeführt, die alle Fahrzeuge zu Dreiecken vernetzt. Die daraus resultierende Netzstruktur wird anschließend auf Fahrzeugschlangen reduziert, sodass nur noch Fahrzeuge miteinander vernetzt sind, die sich auf derselben Fahrbahn befinden. Dies ermöglicht eine Betrachtung der Nachbarfahrzeuge, die direkt davor oder dahinter in Fahrtrichtung existieren. Die finale Unterteilung wird mithilfe von Fuzzy-Logik realisiert. Hierfür ist es erforderlich die Fahrzeuge ausfindig zu machen, die in der Nähe einer Straßenkreuzung und Ampel vorliegen. Da die Straßenkreuzungen nicht bekannt sind, werden diese mithilfe des Verfahrens Template Matching lokalisiert und die dazugehörigen Umkreisbereiche definiert. Die Radiusgröße der Umkreise ist abhängig von der Fahrspurkategorie des jeweiligen Straßenabschnittes. Diese gibt Auskunft über die Anzahl der Fahrbahnen auf einem Straßenabschnitt an. Im Anschluss daran wird geprüft, ob die Fahrzeuge in diesen Bereichen liegen.

Abschließend erfolgt die Unterteilung der Fahrzeuge in „parkend“ und in „am Verkehr teilnehmend“. Hierfür werden die generierten Fahrzeugschlangen und die Fahrzeuge ohne Nachbarschaft separat in einem Fuzzy-System betrachtet. Innerhalb eines Systems werden anhand von vorweg aufgestellten Regeln Sachverhalte miteinander verknüpft (Bothe, 1995, S.1). Dies umfasst die Fahrgeschwindigkeit, die Fahrzeugdichte bei einer Fahrzeugschlange, der Umkreis von Ampeln oder Straßenkreuzungen und die Straßenkategorie.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einführung

1.1. Motivation

Am Institut für Methodik der Fernerkundung des Deutschen Zentrums für Luft- und Raumfahrt in Oberpfaffenhofen werden für das Projekt VABENE++ Methoden für eine effiziente Umsetzung notwendiger Rettungslogistik bei Katastrophen und Großveranstaltungen entwickelt. Ein Teilbereich des Projektes ist das Verkehrsmonitoring und die Lageerfassung. Hierbei werden während eines Fluges Verkehrsdaten aus Bildfrequenzen in Echtzeit extrahiert und anschließend zu einer mobilen Bodenstation übertragen (Leitloff et.al., 2014). Anhand dieser Daten ist im Falle einer Katastrophe oder Großveranstaltung eine sofortige Darstellung der aktuellen Verkehrslage möglich. Ebenso kann bei einer anstehenden Evakuierung überprüft werden, welche Straßen noch befahrbar sind.

Das VABENE-System liefert in Echtzeit die Position von Fahrzeugen mit der dazugehörigen Fahrtrichtung und Fahrgeschwindigkeit. Zu Beginn eines Einsatzes wird das Kamerasystem am Flugzeug montiert und das System hochgefahren. Am Zielgebiet angekommen wird die Kamera vom Operator gestartet. Die Luftbildaufnahme erfolgt mit dem vom DLR entwickelten 4k-Kamerasystem und wird im Burst-Mode realisiert. Neben der Luftbilderfassung werden die Fluglage und die GPS-Position registriert, sodass unter Zuhilfenahme eines Digitalen Oberflächenmodells jedem Pixel im Bild eine Weltkoordinate zugeordnet werden kann. Innerhalb eines Bursts, der aus einer Serie von drei Bildern besteht, werden alle im Bild zu sehenden Fahrzeuge detektiert und getrackt. Die Fahrzeugdetektion erfolgt bei jedem ersten Bild und das Fahrzeugtracking bezieht sich immer auf ein Bildpaar innerhalb eines Bursts. Die Aufnahme eines Bursts erfolgt alle sieben Sekunden. Mithilfe eines Mikrowellenlinks werden diese Daten anschließend zu einer mobilen Bodenstation übertragen. Damit die Station im Zielgebiet nicht mehr zwingend anwesend sein muss, ist ein LTE-Datenlink geplant. Dieser überträgt die Daten auf einem Server im DLR und ermöglicht somit eine direkte

Lokalisierung der Gefahrensituation. Die an der Bodenstation empfangenen Daten werden den Nutzern dann über Internetportale zur Verfügung gestellt. Typische Nutzer sind BOS-Kräfte wie beispielsweise die Polizei.

Damit das VABENE-System neben einer Verkehrssimulation auch eine Betrachtung der Parkraumbelegung innerhalb der Großstädte liefern kann, ist eine Identifizierung aller parkenden Fahrzeuge unerlässlich. Unter Zuhilfenahme der Fahrgeschwindigkeit wird bereits eine Trennung zwischen den stehenden und fahrenden Fahrzeugen realisiert. Allerdings kann nicht davon ausgegangen werden, dass alle stehenden Fahrzeuge parken, da diese ebenso am Verkehr teilnehmen können. Grund für den Stillstand kann in diesem Fall das Warten an einer Ampel, Straßenkreuzung oder ein Stau sein. Um auch zukünftig zwischen diesen beiden Klassen differenzieren zu können, ist eine weitere Methodik erforderlich, welche im Rahmen dieser Bachelorarbeit entwickelt wird.

1.2. Zielsetzung

Das Ziel dieser Arbeit liegt in der Entwicklung einer Methodik, die alle stehenden Fahrzeuge ermittelt und anschließend in parkende und am Verkehr teilnehmende unterteilt. Als Datengrundlage dient das Prozessier Ergebnis der automatischen Verkehrserfassung des DLR. Die Entwicklung des Algorithmus erfolgt mit der Programmiersprache C/C++ unter Windows.

2. Datengrundlage

2.1. Luftbilder

Die Bilder wurden während eines Helikopterfluges mit einem vom DLR eigens entwickelten 4k-Kamerasystem aufgenommen (siehe Abbildung 1). Das Sensorsystem besteht aus drei Canon Kameras, eine Nadir-Kamera und zwei Seitblickkameras.

Abbildung 1: Kamerasystem an der EC 135 des DLR (Luftbildkamerasystem, 2017)

Darüber hinaus verfügt das System über eine GNSS/IMU Einheit, drei Computer für die Onboard-Prozessierung in Real-Time und einen Daten-Downlink (C-Band) (Kurz, et.al., 2012). Der Flug ereignete sich Ende März 2017 über München. Die Aufnahmen wurden hierbei im Burst-Mode getätigt, dieser umfasst drei Bilder. Innerhalb eines Bursts werden die Fahrzeugdetektion und das Tracking realisiert. In Abbildung 2 ist jeweils ein Bildausschnitt aus den drei Luftbildern dargestellt. Das markierte Fahrzeug wurde im linken Bild detektiert und im mittleren und rechten Bild jeweils getrackt. Die Bilder wurden in einem Zeitabstand von 0,8 Sekunden getätigt. Die Auflösung der Bilder beträgt jeweils 10cm.

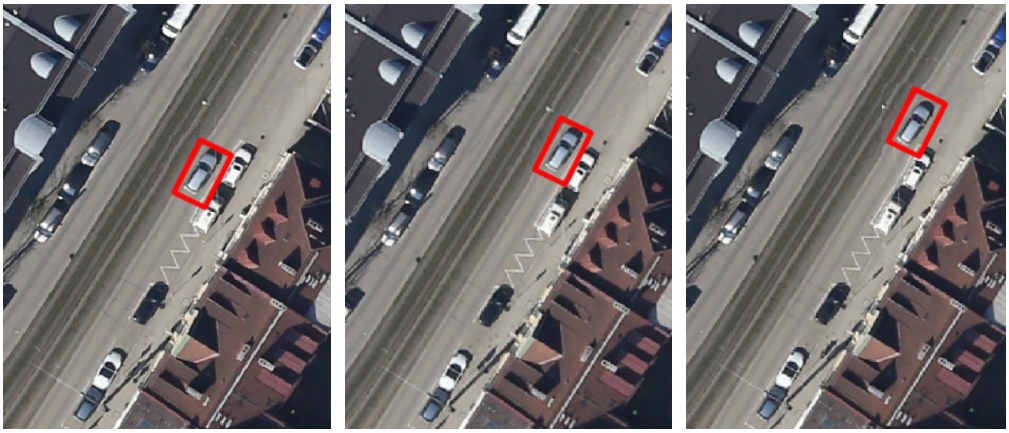


Abbildung 2: Drei Luftbilder eines Bursts

2.2. Fahrzeugdetektion

Die Prozessierung zur automatischen Verkehrserfassung wurde für diese Arbeit im Postprocessing durchgeführt. Das Ergebnis der Prozessierung wird in Form einzelner Dateien dokumentiert und so den Anwendern zur Verfügung gestellt. Bei der Fahrzeugdetektion werden alle im Bild zu sehenden Fahrzeuge detektiert und in der car_detected Datei festgehalten. Diese Datei liegt für jedes Bild vor. Die wichtigsten Informationen hiervon sind die Fahrzeugpositionen. Diese sind neben dem Bildkoordinatensystem auch im UTM Koordinatensystem angegeben.

2.3. Fahrzeugtracking

Das Fahrzeugtracking erstrebt alle detektierten Fahrzeuge im nächsten Bild innerhalb eines Bursts wiederzufinden. Das Ergebnis liefert die _cpdb Datei, die für jedes Tracking einmal und somit für jeden Burst zweimal vorhanden ist. In dieser Datei sind die Fahrzeugpositionen von der Detektion und dem Tracking gegenübergestellt (erstes Tracking) bzw. die Fahrzeugpositionen des Trackings im mittleren Bild denen des Trackings im letzten Bild (zweites Tracking innerhalb des Bursts). Die Positionen liegen hierbei ebenso wie bei der Fahrzeugdetektion im Bild- und UTM-Koordinatensystem vor. Darüber hinaus geben die Daten Auskunft über die Qualität des Trackingvorgangs. Infolgedessen kann eine Berechnung der zurückgelegten Strecke, der Fahrgeschwindigkeit und der Fahrtrichtung durchgeführt werden.

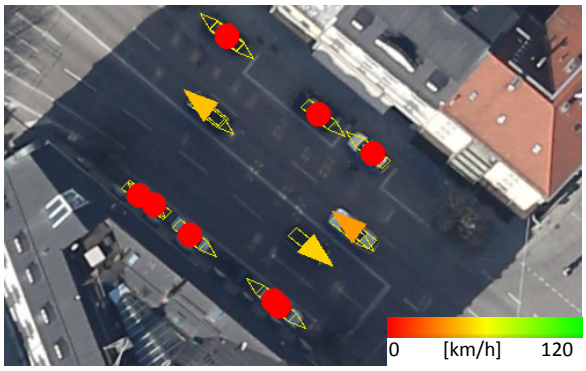


Abbildung 3: Fahrzeugtracking

In Abbildung 3 ist das Ergebnis der automatischen Verkehrserfassung dargestellt. Die Fahrzeuge werden anhand ihrer Geschwindigkeit unterschiedlich farblich dargestellt. Die Pfeile deuten die jeweilige Fahrtrichtung an. Alle Fahrzeuge mit einem roten Punkt weisen gar keine oder nur eine geringe Geschwindigkeit auf und besitzen aus diesem Grund keine Fahrtrichtung.

Um einen schnellen Zugriff zu diesen Informationen bei der Programmierung zu ermöglichen, wird eine Objektstruktur für die Fahrzeuge angelegt. Mithilfe dieser Struktur wird jedes Fahrzeug zum Objekt, welchem Eigenschaften übergeben werden. Jedes Fahrzeug besitzt eine eindeutige ID. Mithilfe der Eigenschaft „tracked“ kann dem Fahrzeug übermittelt werden, ob neben der Detektion auch ein Tracking erfolgte. Eine weitere wichtige Eigenschaft ist die Position des Fahrzeuges, die im UTM- und Bildkoordinatensystem vorliegt. Im weiteren Programmverlauf wird die Struktur um weitere Eigenschaften ergänzt.

struct VEHICLE
int ID
int tracked
int x
int y
double E
double N

Abbildung 4: Struktur Fahrzeug

2.4. NAVTEQ Daten

Bei der automatischen Verkehrserfassung wird auf die NAVTEQ Datenbank zugegriffen und für jeden Burst eine Datei mit dem Namen navteq_roads erstellt. NAVTEQ Daten bieten eine detaillierte Darstellung des Streckennetzes und sind somit Basis für viele Navigationsdienste und Anwendungen (NAVTEQ, 2017).

Abbildung 5: NAVTEQ Linien

Eine NAVTEQ Linie stellt nahezu die Mittelachse der Straße dar und bezieht sich immer auf das erste Bild eines Bursts (siehe Abbildung 5). Nach jeder Straßenkreuzung oder bei einer Änderung der zulässigen Fahrgeschwindigkeit beginnt ein neuer Straßenabschnitt. Jede Linie besteht aus mehreren Verknüpfungspunkten. Die Anreihung der Verknüpfungspunkte repräsentiert die auf dem Straßenabschnitt vorgegebene Fahrtrichtung. Für Straßen, die in beiden Richtungen befahren werden können, existieren zwei NAVTEQ Linien, sodass eine Überlagerung von mehreren Linien gegeben ist (Leitloff et.al., 2014).

In der navteq_roads Datei stehen die UTM-Koordinaten der Linienstützpunkte. Da jede NAVTEQ Linie eine eindeutige ID besitzt, ist eine eindeutige Zuordnung der Stützpunkte zu der zugehörigen NAVTEQ Linie möglich.

Die NAVTEQ Linien werden ebenso wie die Fahrzeuge mithilfe einer Struktur abgespeichert (siehe Abbildung 6). Für den nachfolgenden Programmverlauf werden neben den Koordinaten, auch die jeweilige NAVTEQ ID des Stützpunktes zugänglich gemacht. Weitere Informationen sind die Straßenkategorie und die Fahrspurkategorie. Die Straßenkategorie gibt Auskunft über die Funktion des Straßenabschnittes und die Fahrspurkategorie repräsentiert die Anzahl der Fahrbahnen (Digital Map Technology, 2017).

struct NAVTEQ
int id
int x
int y
double E
double N
int road_category
int lane_category

Abbildung 6: Struktur NAVTEQ

2.5. Ampeldaten

Für diese Arbeit steht zusätzlich ein Datensatz mit Ampelpositionen für den Großraum München zur Verfügung. Die Daten befinden sich im UTM-Koordinatensystem. Für die Arbeit mit Bildern ist allerdings ein Übergang ins Bildkoordinatensystem erforderlich. Aus diesem Grund werden die Koordinaten mithilfe des georeferenzierten Bildes und dessen Auflösung vom UTM- ins Bildkoordinatensystem umgerechnet. Für den weiteren Programmverlauf ist es wichtig nur die Ampelpositionen zu verwenden, die im Bildbereich liegen.

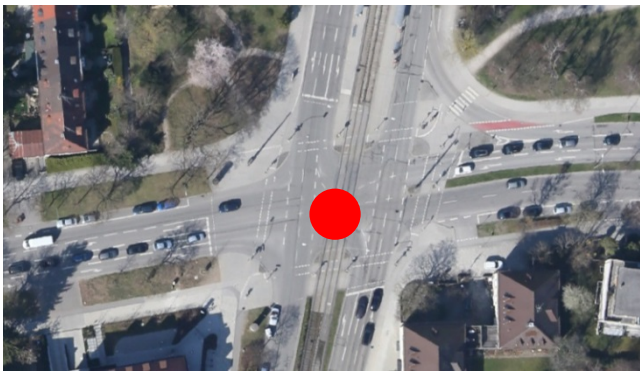


Abbildung 7: Ampelposition

In Abbildung 7 ist ein Ausschnitt eines Luftbildes mit einer Ampel dargestellt. Für diese Daten wird ebenfalls eine Objektstruktur angelegt, sodass ein simples Abgreifen der Koordinaten ermöglicht wird (siehe Abbildung 8).

struct TRAFFICLIGHT
int id
int x
int y
double E
double N

Abbildung 8: Struktur Ampel

3. Methodik

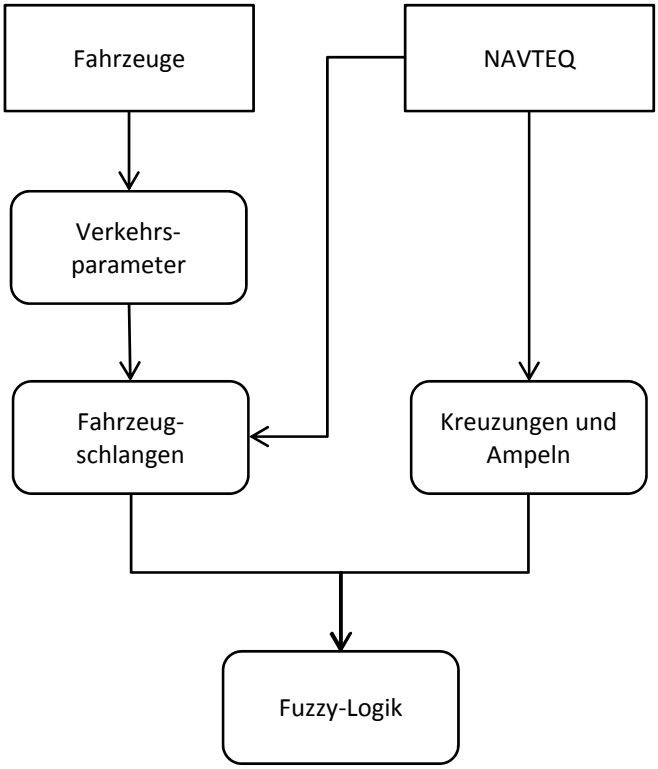


Abbildung 9: Programmablauf - Methodik

Diese Methodik realisiert eine Differenzierung zwischen den am Verkehr teilnehmenden und den parkenden Fahrzeugen. Von jedem getrackten Fahrzeug liegen zwei Positionen vor, sodass eine Berechnung von Verkehrsparametern, wie beispielsweise die zurückgelegte Strecke, Fahrgeschwindigkeit und Fahrtrichtung, durchgeführt werden kann.

Die Unterscheidung erfolgt anschließend vorrangig unter Betrachtung der Nachbarfahrzeuge. Aus diesem Grund werden auf Basis der Fahrzeugpositionen einzelne Fahrzeugschlangen generiert. Eine Fahrzeugschlange setzt sich aus benachbarten Fahrzeugen zusammen, die sich auf demselben Straßenabschnitt befinden. Darüber hinaus befindet sich jede Schlange parallel zur Mittelachse der Straße. Ebenso verfügen die teilnehmenden Fahrzeuge über dieselbe Fahrtrichtung.

Daraufhin wird für jedes Fahrzeug untersucht, ob im nahen Umkreis eine Ampel oder Straßenkreuzung existiert. Im Gegenzug zu den Ampeln, deren Koordinaten bereits in einer Datei vorliegen, müssen die Straßenkreuzungen erstmalig lokalisiert werden.

Im Anschluss daran wird unter Verwendung von Fuzzy-Logik eine Unterscheidung in am Verkehr teilnehmende und parkende Fahrzeuge vollzogen. Hierbei wird jeweils ein System für die Fahrzeugschlangen und für die Fahrzeuge ohne Nachbarschaft aufgebaut. Innerhalb eines Systems erfolgt neben der Fahrgeschwindigkeit auch eine Betrachtung der Umgebungssituation. Die individuell herrschenden Sachverhalte werden hierbei mithilfe von vorab definierten Regeln miteinander verknüpft und fließen somit in die finale Bewertung mit ein (Bothe, 1995, S.1).

3.1. Verkehrsparameter

Für die Unterteilung in parkende und am Verkehr teilnehmende Fahrzeuge ist die Ermittlung von Verkehrsparametern erforderlich. Mithilfe der automatischen Verkehrserfassung wurden die Fahrzeuge detektiert und anschließend getrackt. Von allen getrackten Fahrzeugen liegen somit zwei Positionen vor. Dadurch ist eine Berechnung der zurückgelegten Strecke, der Fahrgeschwindigkeit und der Fahrtrichtung durchführbar.

Mithilfe der für die Fahrzeuge vorab angelegten Struktur VEHICLE werden diese berechneten Parameter jedem Fahrzeug als Eigenschaft übergeben und stehen somit für den weiteren Programmverlauf zur Verfügung (siehe Abbildung 10).

struct VEHICLE
int ID
int tracked
int x
int y
double E
double N
double driving_distance
double speed
double driving_direction

Abbildung 10: Struktur Fahrzeug - Verkehrsparameter

3.1.1. Zurückgelegte Strecke

Dieser Verkehrsparameter gibt Auskunft über den Weg, den das Fahrzeug zwischen der Detektion und dem Tracking zurückgelegt hat. Die Bestimmung ist Voraussetzung für die nachfolgende Berechnung der Fahrgeschwindigkeit. In der untenstehenden Abbildung werden die detektierte und die getrackte Position des Fahrzeuges mit den grünen Pfeilen veranschaulicht. Die gestrichelte Linie stellt die zurückgelegte Strecke dar.

Abbildung 11: Zurückgelegte Strecke

Für die Berechnung wird der Satz des Pythagoras angewandt. Als Datengrundlage dienen hierfür die UTM Koordinaten des Fahrzeuges von der Detektion und vom Tracking.

$$s_{1^2} = \sqrt{\Delta y^2 + \Delta x^2}$$

s_1^2 Strecke von Punkt1 nach Punkt2

Δy, Δx Koordinatendifferenzen

(Gruber & Joeckel, 2014, S. 40)

3.1.2. Fahrgeschwindigkeit

Unter Kenntnisnahme der Fahrgeschwindigkeit ist eine Abgrenzung aller stehenden Fahrzeuge möglich. Für die Berechnung ist ein Bezug zu den Metadaten der einzelnen Bilder notwendig. Aus diesem Grund werden aus den aux-Dateien der ersten zwei Bilder eines Bursts die Werte der „GPSIMU.WeekSec“ entnommen und davon die Differenz gebildet. Das Ergebnis entspricht der Zeitdifferenz zwischen der Detektion und dem Tracking in Sekunden. Im Anschluss daran kann zusammen mit der zurückgelegten Strecke aus dem Kapitel 3.1.1. eine Berechnung der Fahrgeschwindigkeit vorgenommen werden. Hierbei wird die Strecke durch die Zeit dividiert.

$$v = \frac{s}{t}$$

v Geschwindigkeit

s Strecke

t Zeit

(Formelsammlung, 2017)

3.1.3. Fahrtrichtung

Für die Identifizierung von Fahrzeugschlangen ist ein Bezug zur Fahrtrichtung aller getrackten Fahrzeuge erforderlich. Für die Bestimmung wird der Richtungswinkel von der Position der Detektion zu der Position des Trackings berechnet.

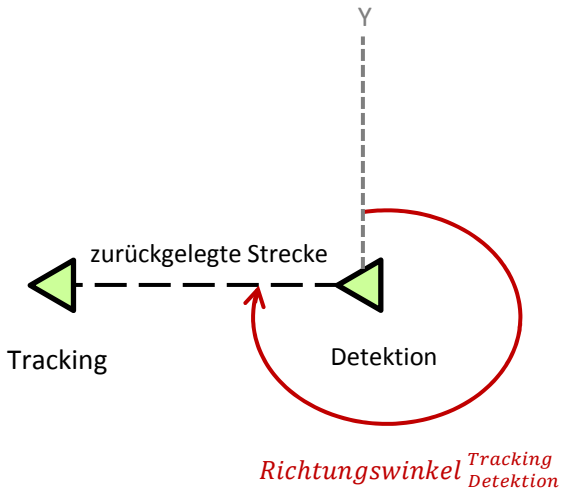


Abbildung 12: Fahrtrichtung

In Abbildung 12 ist die Situation eines getrackten Fahrzeuges im Bildkoordinatensystem dargestellt. Da das Fahrzeug detektiert und getrackt wurde besitzt es zwei Positionen. Der Richtungswinkel von der detektierten zur getrackten Position ist die Fahrtrichtung. Dieser wird in der Abbildung mittels grüner Pfeile dargestellt. Für die Berechnung kommt der Arkustangens zur Anwendung.

$$t_{1^2} = \arctan(\frac{\Delta y}{\Delta x})$$

t_1^2 Richtungswinkel von Punkt 1 nach Punkt 2

Δy, Δx Koordinatendifferenzen

(Gruber & Joeckel, 2014, S. 39)

3.2. Identifizierung von Fahrzeugschlangen

Für die Unterscheidung in parkende und am Verkehr teilnehmende Fahrzeuge sind Informationen über die benachbarten Fahrzeuge erforderlich. Um einen Bezug zur Nachbarschaft herzustellen, werden einzelne Fahrzeugschlangen identifiziert. Eine Fahrzeugschlange beinhaltet nur Fahrzeuge, die sich auf demselben Straßenabschnitt und auf derselben Fahrbahn befinden. Der Programmablauf wird in Abbildung 13 veranschaulicht.

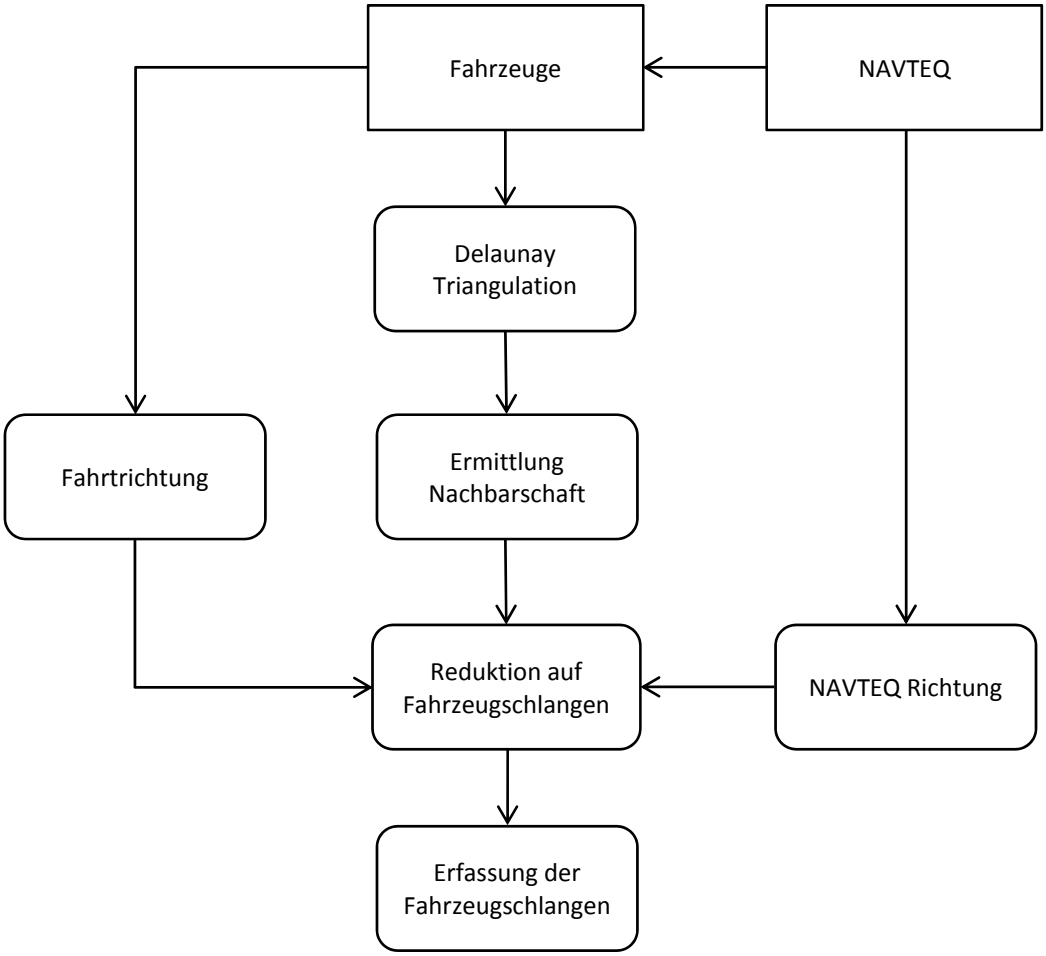


Abbildung 13: Programmablauf - Identifizierung von Fahrzeugschlangen

Für die Generierung der Fahrzeugschlangen werden für jedes Fahrzeug die Straßenabschnitte ermittelt, auf denen es sich befindet. Da sich gelegentlich NAVTEQ Linien überlagern, können einem Fahrzeug mehrere Straßenabschnitte zugeordnet werden. Nachfolgend wird eine Delaunay

Triangulation durchgeführt. Das Ergebnis ist ein Graph, dessen Knoten die Fahrzeuge repräsentiert und die Kanten eine Verbindung zwischen den benachbarten Fahrzeugen herstellen. Mithilfe des Delaunay Graphs lassen sich für alle Fahrzeug die benachbarten Fahrzeuge abgreifen (Knöttner, 2017, S. 8).

Um anschließend eine gezielte Betrachtung der Nachbarschaft auf derselben Fahrbahn gewährleisten zu können, wird der Graph auf Fahrzeugschlangen reduziert. Hierfür werden Bedingungen aufgestellt, die jede Kante des Graphs erfüllen muss. Dies umfasst die NAVTEQ ID, die NAVTEQ Richtung, die Fahrtrichtung und die Anzahl der Nachbarn. Darüber hinaus darf ein Fahrzeug nur einmal Nachbarfahrzeug sein. Falls eine Bedingung nicht zutrifft, wird die Kante und somit das Nachbarfahrzeug aus der Netzstruktur entfernt. Im Anschluss daran werden die Fahrzeuge innerhalb einer Fahrzeugschlange ermittelt und gesondert abgespeichert.

3.2.1. Delaunay Triangulation

Für die Herstellung von Nachbarschaftsbeziehungen zwischen den Fahrzeugen untereinander wird eine Delaunay Triangulation durchgeführt. Die Triangulation wird mithilfe der Programmbibliothek OpenCV vollzogen. Mithilfe der Delaunay Triangulation werden alle Punkte zu Dreiecken mit möglichst großen Innenwinkeln vernetzt. Darüber hinaus darf sich innerhalb eines Dreiecks kein weiterer Punkt befinden, was eine Vernetzung benachbarter Fahrzeuge bewirkt. Das Ergebnis der Triangulation ist ein Graph, dessen Bestandteile Knoten und Kanten sind (Bradski & Kaehler, 2017, S.923). In diesem Anwendungsfall sind die Fahrzeugpositionen die Knoten und die Verbindungslinien zwischen benachbarter Fahrzeuge die Kanten (siehe Abbildung 14).

Abbildung 14: Delaunay Triangulation

3.2.2. Ermittlung der Nachbarschaft

Mithilfe der Kanten des Delaunay Graphs ist für jedes Fahrzeug das Abgreifen der benachbarten Fahrzeuge möglich (Knöttner, 2017, S. 18). Jede Kante besteht aus einem Anfangs- und einem Endpunkt, die durch Fahrzeuge verkörpert werden. Ein Beispiel wird in Abbildung 15 veranschaulicht. Die Fahrzeuge werden als grüne Pfeile repräsentiert. Die rote Verbindungslinie zwischen den beiden Fahrzeugen ist die Kante.

Abbildung 15: Kante aus dem Delaunay Graph

Zu Beginn wird ermittelt, um welche Fahrzeuge es sich bei den Kantenendpunkten handelt. Im Anschluss daran können sich die beiden Fahrzeuge gegenseitig als Nachbarn hinzufügen. Unter Kenntnisnahme der Nachbarfahrzeuge erfolgt eine Berechnung der Kantenlänge und der Kantenrichtung. Die Kantenlänge ist der Abstand zwischen den beiden Fahrzeugen. Die Kantenrichtung entspricht dem Richtungswinkel vom Fahrzeug zum Nachbarfahrzeug. Mithilfe des Winkels kann im späteren Verlauf überprüft werden, ob sich das Nachbarfahrzeug direkt in Fahrtrichtung vor oder hinter dem Fahrzeug befindet. Hierfür werden die Formeln aus Kapitel 3.1.1 und 3.1.3 verwendet.

struct NEIGHBOR
int Vehicle_ID double distance double angle

Abbildung 16: Struktur Nachbarfahrzeug

Um einen schnellen Zugriff zu den Nachbarfahrzeugen ermöglichen zu können, wird speziell für die Nachbarfahrzeuge eine gesonderte Struktur angelegt. Mithilfe dieser Struktur wird jedes Nachbarfahrzeug zu einem Objekt (siehe Abbildung 16). Anhand der Fahrzeug ID ist jederzeit ein Zugriff auf die Fahrzeugeigenschaften des Nachbarfahrzeuges, wie beispielsweise die Koordinaten oder die Fahrgeschwindigkeit, möglich. Darüber hinaus wird dem Fahrzeug die vorab berechnete Kantenrichtung und Kantenlänge als Eigenschaft mitgegeben.

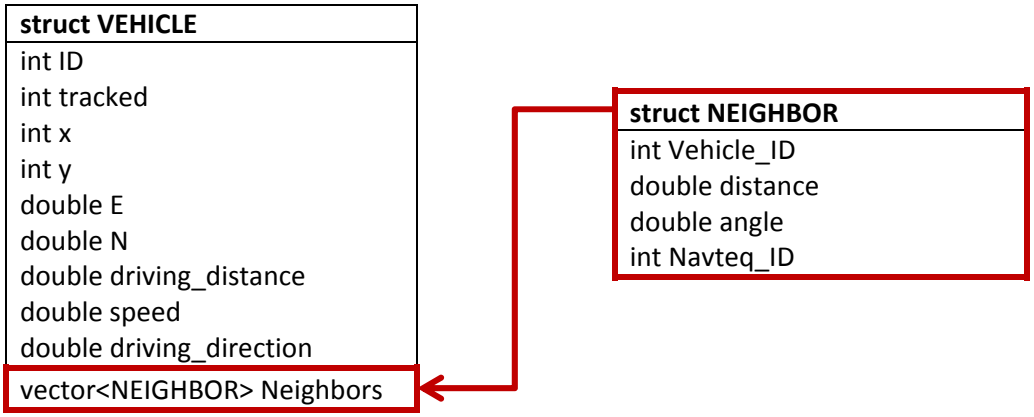


Abbildung 17: Nachbarschaft eines Fahrzeuges

Die Nachbarfahrzeuge werden anschließend einem Vektor hinzugefügt und unter Zuhilfenahme der Objektstruktur dem Fahrzeug als Eigenschaft übergeben (siehe Abbildung 17).

3.2.3. Ermittlung der NAVTEQ ID

Für die Identifizierung von Fahrzeugschlangen auf einem Straßenabschnitt ist ein Bezug zu den Straßenabschnitten Voraussetzung. Jede NAVTEQ Linie verfügt über eine eindeutige NAVTEQ ID. Die Stützpunkte der Navteq Linien liegen mit der zugehörigen ID in einem Vektor vor. Der Programmablauf ist in Abbildung 18 dargestellt. Für jedes Fahrzeug werden folglich die Straßenabschnitte ermittelt, auf denen es sich befindet.

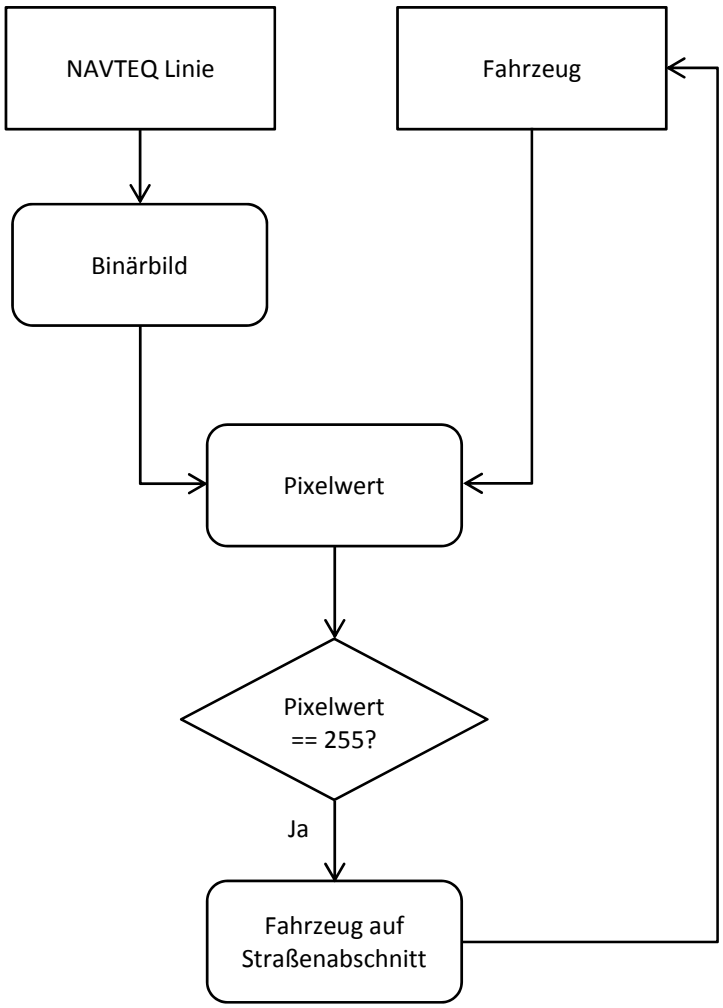


Abbildung 18: Programmablauf - Abgreifen der NAVTEQ ID

Für die Ermittlung der zugehörigen NAVTEQ IDs eines Fahrzeuges wird von jedem Straßenabschnitt ein gesondertes Binärbild erstellt. Im Bild erhält die NAVTEQ Linie den Pixelwert 255, wobei der Rest

den Pixelwert 0 besitzt. Für die Annäherung an die Straßenbreite, erhält die Linie eine Breite 22m. Diese Größe wurde ebenso bei der automatischen Verkehrserfassung des DLR verwendet und gewährleistet eine Abdeckung aller innerstädtischen Straßenarten (Leitloff et.al., 2014).

Nach der Erzeugung des Binärbildes pro Straßenabschnitt wird für jede Fahrzeugposition der Pixelwert im Bild abgefragt. Falls der Pixelwert 255 beträgt, befindet sich das Fahrzeug auf diesem Straßenabschnitt und die jeweilige NAVTEQ ID wird mithilfe der vorab angelegten Struktur dem Fahrzeug als Eigenschaft übergeben (siehe Abbildung 19).

struct VEHICLE
int ID
int tracked
int x
int y
double E
double N
double driving_distance
double speed
double driving_direction
vector<NEIGHBOR> Neighbors
vector<int> NAVTEQ_ID

Abbildung 19: Struktur Fahrzeug - NAVTEQ-ID

Da sich einige Straßenabschnitte überlagern, wurde jedem Fahrzeug die Möglichkeit eingeräumt mehrere NAVTEQ IDs mitzuführen.

3.2.4. Bestimmung der NAVTEQ Richtung

Für die Reduktion des Delaunay Graphs auf Fahrzeugschlangen ist die Ermittlung der NAVTEQ Richtung erforderlich. Um lediglich eine Verkantung von hintereinander fahrenden Fahrzeugen zu erhalten, muss die Kantenrichtung mit der vorgegebenen Fahrtrichtung auf dem gegenwärtigen Straßenabschnitt übereinstimmen. Diese NAVTEQ Richtung wird auf Basis der Stützpunkte ermittelt.

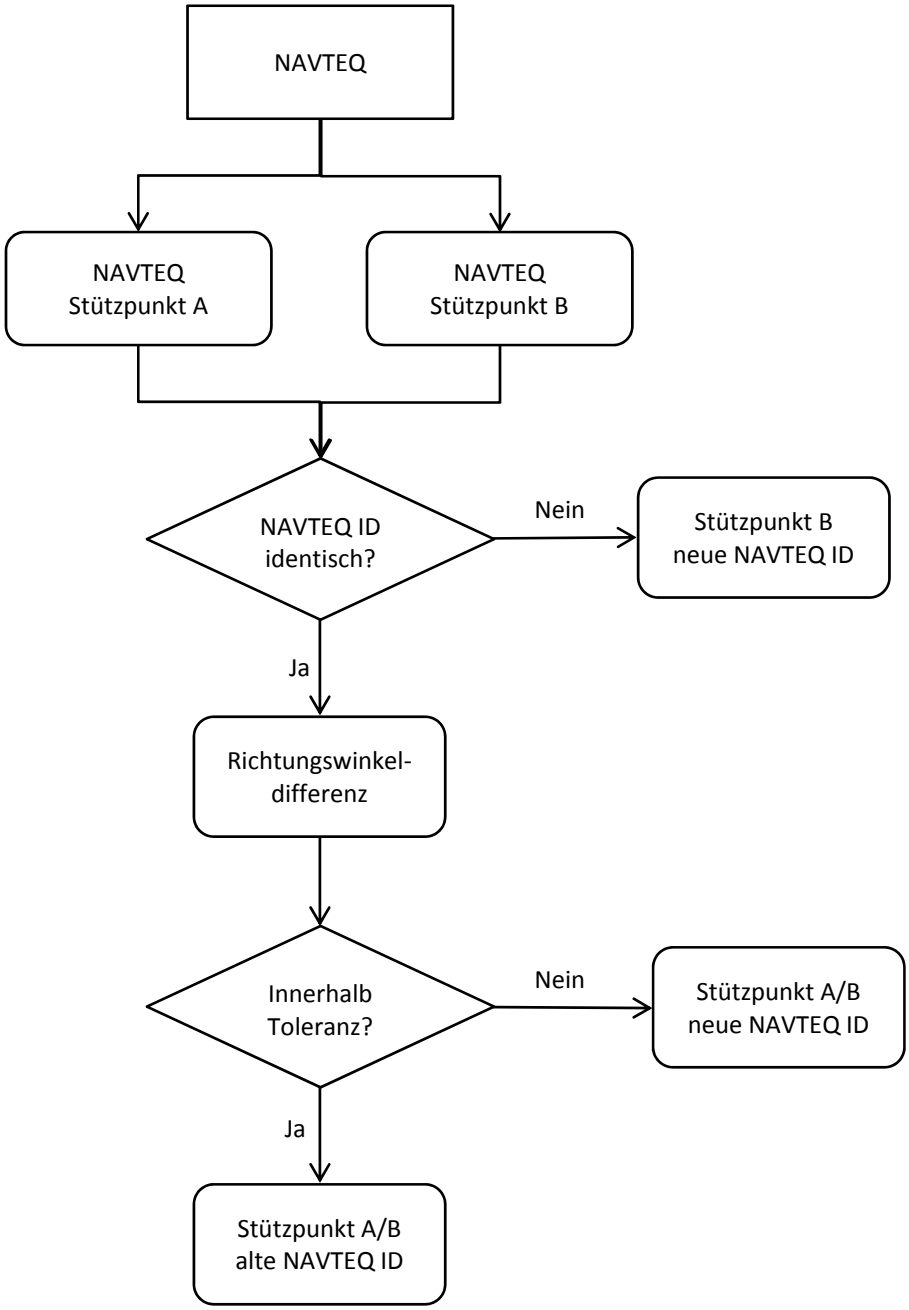


Abbildung 20: Programmablauf – NAVTEQ ID Vergabe

Da allerdings nicht alle NAVTEQ Linien ausschließlich über einen geraden Verlauf verfügen, muss vorab die Linie bei jedem Richtungswechsel geteilt werden. Um dies zu erreichen, erhalten alle Stützpunkte eine neue NAVTEQ ID. Der Programmablauf wird in Abbildung 20 veranschaulicht. Paarweise wird überprüft, ob die Stützpunkte dieselbe NAVTEQ ID besitzen und somit demselben Straßenabschnitt angehören. Ist dies nicht der Fall erhält der zweite Stützpunkt eine neue ID.

Falls die beiden Stützpunkte demselben Straßenabschnitt angehören, wird ein Richtungswinkel vom ersten zum zweiten Punkt berechnet. Dieser Richtungswinkel wird mit dem Winkel aus dem vorherigen Stützpunktepaar derselben NAVTEQ ID verglichen. Als Toleranzwert wurden 5gon verwendet. Falls die Differenz außerhalb der Toleranz liegt, wird eine neue NAVTEQ ID vergeben. In diesem Fall liegt eine Richtungsänderung auf dem Straßenabschnitt vor und die Linie muss geteilt werden. Liegt der Wert innerhalb der Toleranz erhalten die Punkte die ID des Vorgängerpaares. Die daraus neu entstandenen NAVTEQ IDs wurden bereits beim Abgreifen der NAVTEQ ID im Kapitel 3.2.3 berücksichtigt.

Nach der Teilung ist die Ermittlung der vorgegebenen Fahrtrichtung auf jeden Straßenabschnitt möglich. Für jedes Stützpunktepaar innerhalb derselben NAVTEQ ID wird ein Richtungswinkel gemäß Kapitel 3.1.3 berechnet. Im Anschluss daran wird das arithmetische Mittel gebildet und als endgültige NAVTEQ Richtung für den Straßenabschnitt verwendet.

Für das effektive Arbeiten mit dem Winkel im weiteren Programmverlauf wird die Struktur „NAVTEQ_ROADS_DIRECTION“ angelegt (siehe Abbildung 21). Mithilfe dieser Struktur wird das Abgreifen der NAVTEQ Richtung für jeden Straßenabschnitt ermöglicht. Darüber hinaus wird die Straßenkategorie der NAVTEQ Linie bei der Objektstruktur für spätere Zwecke mit übernommen.

struct NAVTEQ_ROADS_DIRECTION
int Navteq_ID double Navteq_Direction int Road_Category

Abbildung 21: Struktur Navteqrichtung

3.2.5. Reduktion auf Fahrzeugschlangen

Für die spätere Differenzierung der Fahrzeuge ist die Reduktion des Delaunay Graphs auf Fahrzeugschlangen erforderlich. Das Ziel liegt hierbei in der Verkantung von Fahrzeugen, die sich auf demselben Straßenabschnitt befinden. Darüber hinaus sollen nur Nachbarfahrzeuge existieren, die sich direkt vor oder hinter dem Fahrzeug in Fahrtrichtung befinden. Um dies zu erreichen, müssen Kanten aus dem Delaunay Graph eliminiert werden. Mit dem Ausschluss einzelner Kanten werden Fahrzeugschlangen erzeugt, die sich parallel zur Mittelachse der Straße befinden.

Um den Delaunay Graph auf diese Fahrzeugschlangen reduzieren zu können, werden Kriterien aufgestellt, die jede Kante und somit das Nachbarfahrzeug erfüllen muss:

- Kantenrichtung ≈ NAVTEQ Richtung
- Fahrtrichtung ≈ NAVTEQ Richtung
- Pro NAVTEQ ID ein Nachbarfahrzeug
- Pro NAVTEQ ID ein Zielnachbarfahrzeug

Hierbei entspricht die NAVTEQ Richtung der vorgegebenen Fahrtrichtung auf einem Straßenabschnitt. Diese Richtung wird wie in Kapitel 3.2.4 beschrieben ermittelt. Die Kantenrichtung ist der Richtungswinkel vom Fahrzeug zum Nachbarfahrzeug, welcher nach Kapitel 3.2.2 berechnet wird. Dieser Winkel muss mit der vorgegebenen Fahrtrichtung nahezu übereinstimmen. Ist dies der Fall, befinden sich die Fahrzeuge parallel zur Mittelachse des Straßenabschnittes.

Für die Generierung einzelner Fahrzeugschlangen ist es erforderlich, für jedes Fahrzeug nur ein Nachbarfahrzeug pro NAVTEQ Linie zuzulassen. Ebenso darf ein Fahrzeug nur einmal Ziel- bzw. Nachbarfahrzeug sein. Um dies zu erreichen, muss vorab ermittelt werden, ob das Fahrzeug der Nachbar eines anderen Fahrzeuges ist.

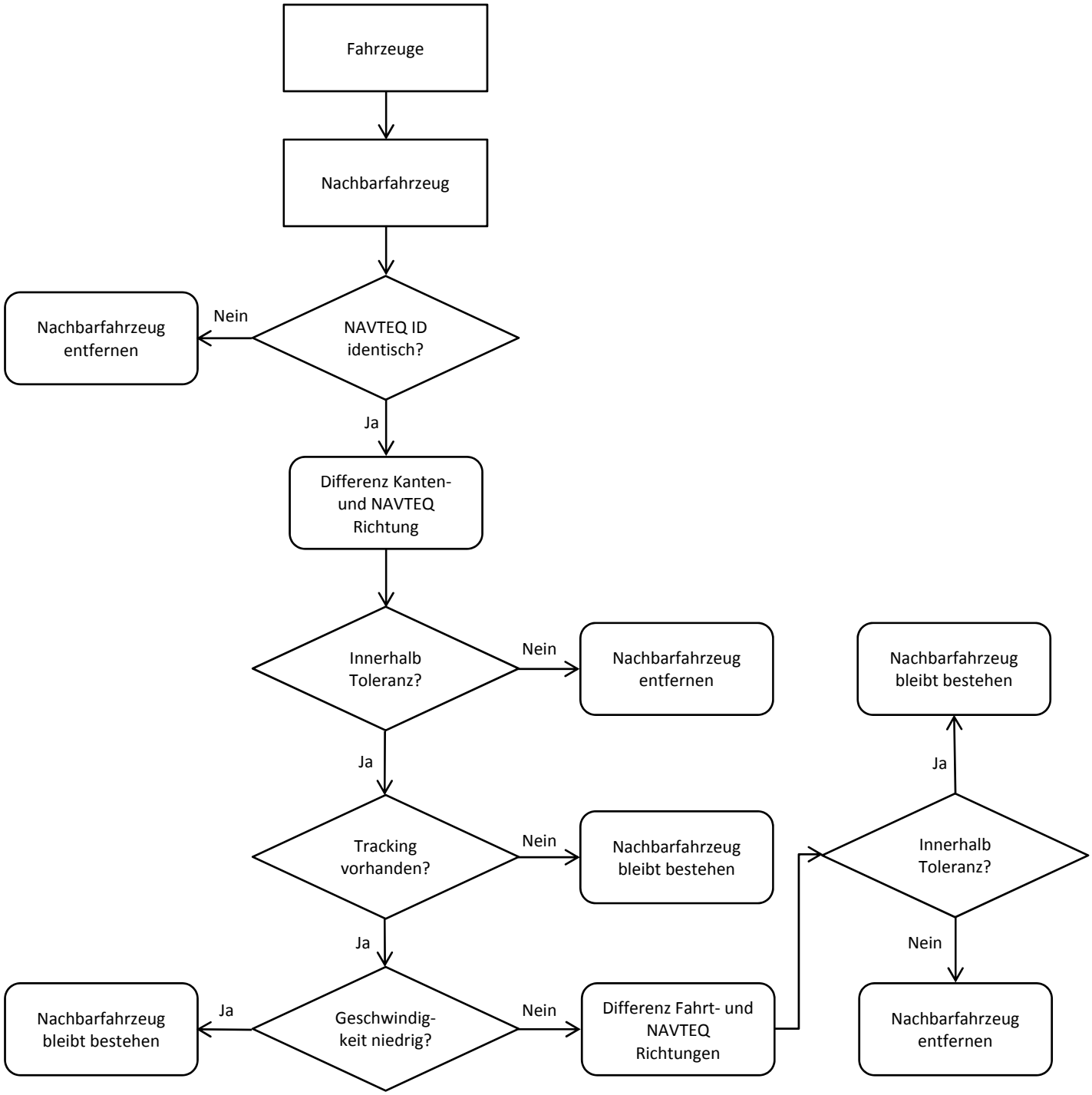


Abbildung 22: Programmablauf - Reduktion auf Fahrzeugschlangen (1)

Die oben stehende Abbildung 22 repräsentiert den ersten Teil des Programmablaufs. Zu Beginn werden von jedem Fahrzeug die Nachbarfahrzeuge abgegriffen und geprüft, ob diese dieselbe NAVTEQ ID besitzen und demzufolge demselben Straßenabschnitt angehören. Wichtig ist hierbei, dass jedes Fahrzeug mehreren Straßenabschnitten angehören und somit mehrere NAVTEQ IDs

besitzen kann. Wurde eine Übereinstimmung gefunden, wird die Differenz zwischen der Kantenrichtung und der NAVTEQ Richtung ermittelt. Mithilfe dieser Differenz kann geprüft werden, ob sich die Fahrzeuge parallel zur Straßenachse befinden. Für diese Überprüfung wird ein Toleranzwert definiert. Wird die Toleranz zu klein gewählt, gehen wichtige Nachbarschaftsbeziehungen verloren. Ist sie zu hoch, bleiben störende Kanten bestehen, die das Endergebnis negativ beeinflussen können. Es ist daher ein Wert erforderlich, bei dem diese Nachteile minimal sind. Ein hierfür geeigneter Wert liegt zwischen 5 und 10 Gon.

Liegt die Winkeldifferenz außerhalb der Toleranz, befindet sich das Nachbarfahrzeug nicht auf derselben Fahrspur und wird daraufhin aus der Nachbarschaft entfernt. Liegt die Differenz innerhalb der Toleranz, wird zusätzlich geprüft, ob das Nachbarfahrzeug neben der Detektion auch getrackt wurde. Wurde das Fahrzeug lediglich detektiert, bleibt die Nachbarschaftsbeziehung bestehen. Liegt allerdings eine getrackte Position des Fahrzeuges vor, wird die Fahrgeschwindigkeit des Nachbarfahrzeuges abgegriffen. Ist die Fahrgeschwindigkeit niedrig und somit kleiner als 5km/h, bleibt die Nachbarschaftsbeziehung ebenfalls erhalten.

Hat das Nachbarfahrzeug jedoch eine mittlere bis höhere Fahrgeschwindigkeit wird eine Differenz zwischen der Fahrtrichtung dem Nachbarfahrzeug mit der NAVTEQ Richtung berechnet. Mithilfe dieser Differenz kann festgestellt werden, ob sich das Nachbarfahrzeug parallel zur Straßenachse bewegt. Hierfür wird erneut ein weiterer Toleranzwert festgelegt. Ein sinnvoller Wert liegt hier ebenso zwischen 5 und 10 Gon. Liegt die Differenz innerhalb dieser Toleranz, bleibt das Fahrzeug als Nachbar erhalten. Andererseits wird die Nachbarschaftsbeziehung entfernt.

Dieser Vorgang wird für alle Fahrzeuge einschließlich ihrer Nachbarfahrzeuge durchgeführt. Nach diesem Programmablauf erfüllen die Kanten allerdings nur den ersten Teil der vorab definierten Kriterien. Die Kantenrichtungen und gegebenfalls die Fahrtrichtungen aller getrackten Nachbarfahrzeuge entsprechen der NAVTEQ Richtung. Um jedoch den restlichen Kriterien ebenso gerecht zu werden, müssen die Nachbarfahrzeuge auf weitere Aspekte geprüft und

dementsprechend eliminiert werden. Die Fortsetzung des Programms wird in Abbildung 23 dargestellt.

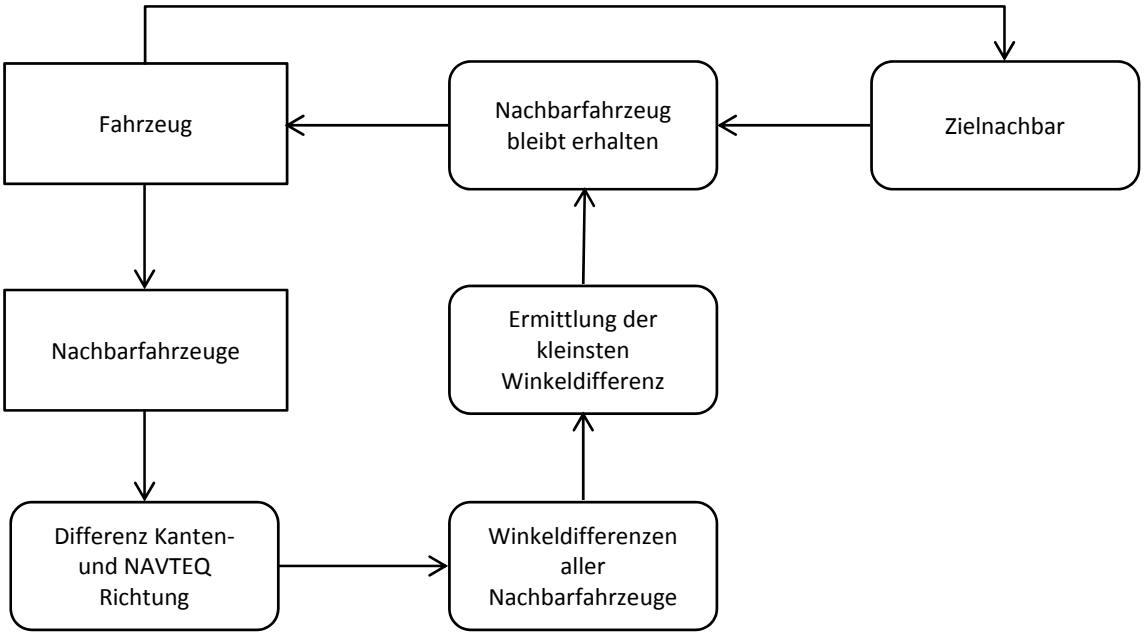


Abbildung 23: Programmablauf - Reduktion auf Fahrzeugschlangen (2)

Das Ziel dieses Programmabschnittes liegt in der Aussortierung der Nachbarschaftsbeziehungen eines Fahrzeuges pro Straßenabschnitt. Für jede NAVTEQ Linie soll anschließend nur noch ein Nachbarfahrzeug existieren. Ein Fahrzeug kann mehreren Straßenabschnitten angehören und somit mehrere NAVTEQ IDs besitzen. Für jede NAVTEQ ID werden die Nachbarfahrzeuge ermittelt, die ebenso diesem Straßenabschnitt angehören. Daraufhin wird die Differenz zwischen der Kantenrichtung und die Richtung dieses Straßenabschnittes ermittelt und zwischengespeichert. Das Resultat ist eine Sammlung von Winkeldifferenzen aller Nachbarfahrzeuge eines Fahrzeuges pro Straßenabschnitt.

Im Anschluss daran wird das Nachbarfahrzeug mit der kleinsten Winkeldifferenz ermittelt. Dieses Nachbarfahrzeug bleibt als einziges dem Fahrzeug als Nachbar erhalten. Alle restlichen Nachbarschaftsbeziehungen auf diesen Straßenabschnitt werden eliminiert. Zugleich wird dem Nachbarfahrzeug über die Struktur VEHICLE mitgeteilt, dass es ein Nachbar ist. Hierbei wird dem

Nachbarfahrzeug die ID des Fahrzeuges hinzugefügt. Dies erfolgt über die Eigenschaft „NeighborFromVehicleID“ (siehe Abbildung 24).

struct VEHICLE
int ID
int tracked
int x
int y
double E
double N
double driving_distance
double speed
double driving_direction
vector<NEIGHBOR> Neighbors
vector<NEIGHBOR> NeighborFromVehicleID

Abbildung 24: Struktur Fahrzeug - Zielnachbar

Mithilfe des neuen Attributes kann für jedes Fahrzeug hinterher geprüft werden, bei welchen Fahrzeugen es Teil der Nachbarschaft ist. Dies ist wichtig für die endgültige Reduktion des Delaunay Graphs auf Fahrzeugschlangen. Der finale Ablauf für die Reduktion auf Fahrzeugschlangen ist in Abbildung 25 dargestellt.

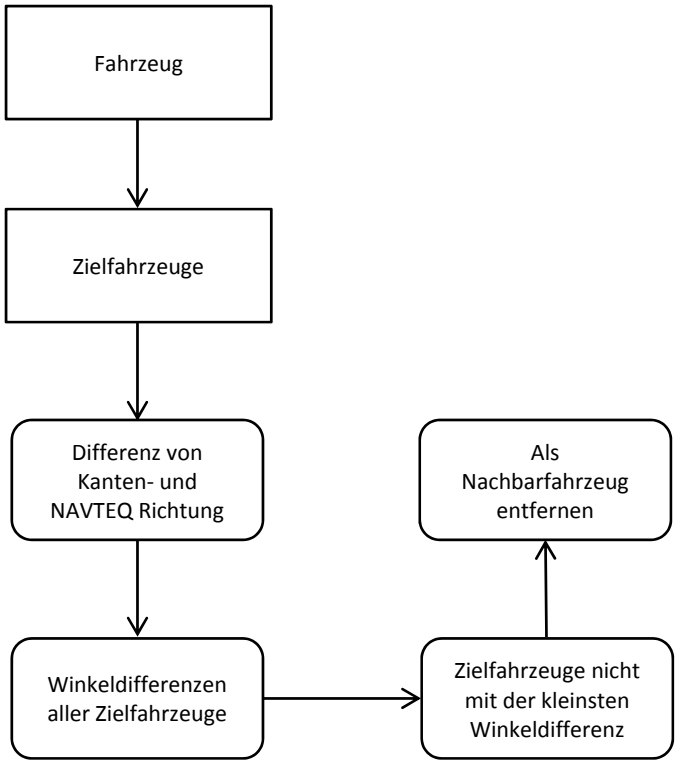


Abbildung 25: Programmablauf - Reduktion auf Fahrzeugschlangen (3)

Von jedem Fahrzeug werden pro NAVTEQ ID die Zielfahrzeuge ermittelt. Ein Zielfahrzeug besitzt das jeweilige Fahrzeug als Nachbar. Von allen Zielfahrzeugen pro Straßenabschnitt werden die Winkeldifferenzen zwischen der Kanten- und NAVTEQ Richtung ermittelt und erneut zwischengespeichert. Hiervon werden alle Zielfahrzeuge ermittelt, die nicht den kleinsten Winkel besitzen. Das Zielfahrzeug mit dem kleinsten Winkel darf als Nachbar eines anderen Fahrzeuges bestehen bleiben. Alle anderen Zielfahrzeuge müssen das Fahrzeug aus der Nachbarschaft entfernen.

In Abbildung 26 sind die Fahrzeugschlangen abgebildet. Die Fahrzeuge sind hierbei anhand ihrer Geschwindigkeit farblich dargestellt. Die detektierten und nicht getrackten Fahrzeuge sind als blauer Punkt markiert. Es wird deutlich, dass sich die Schlangen parallel zur Mittelachse der Straße befinden. Darüber hinaus ist zu erkennen, dass unter den Fahrzeugen Fehldetektionen vorhanden sind. Bei der automatischen Verkehrserfassung wurden auch andere Objekte fälschlicherweise als Fahrzeuge erkannt, wenn diese ähnliche Erscheinungsmerkmale besitzen. In den meisten Fällen sind diese

Fehldetektionen jedoch nicht Teil einer Fahrzeugschlange, da diese überwiegend abseits von der Fahrbahn auftreten.

Abbildung 26: Fahrzeugschlangen

3.2.6. Erfassung der Fahrzeugschlangen

Nachdem die Netzstruktur auf Fahrzeugschlangen reduziert wurde, werden die einzelnen Fahrzeugschlangen aus den Nachbarschaftsbeziehungen abgegriffen und anschließend mittels einer dafür angefertigten Struktur gesondert abgespeichert. In Abbildung 27 ist eine Fahrzeugschlange dargestellt. Es wird ersichtlich, dass für jedes Fahrzeug lediglich ein Nachbar existiert und das Fahrzeug auch nur einmal als Nachbar pro Straßenabschnitt fungiert. Ebenso sind die Kantenlinien nahezu parallel zur Straßenachse.

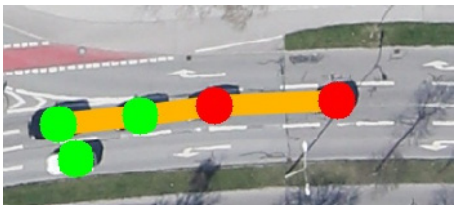


Abbildung 27: Fahrzeugschlange

Beim Betrachten dieses Bildausschnittes wird deutlich, dass mithilfe von Nachbarfahrzeugen eine Ursache für den Stillstand gefunden werden kann. Unter Berücksichtigung der Nachbarfahrzeuge wird deutlich, dass alle am Verkehr beteiligt sind, da die Fahrzeuge am Schlangenanfang fahren. Die Fahrzeuge könnten sich daher an einer Straßenkreuzung befinden, wobei die ersten schon losgefahren sind und der Rest noch wartet.

Die Struktur der Fahrzeugschlangen wird in der untenstehenden Abbildung 28 veranschaulicht. Jede Fahrzeugschlange verfügt über eine eindeutige ID und über einen Vektor mit den teilnehmenden Fahrzeugen. Ein weiterer Bestandteil ist der zugehörige Straßenabschnitt.

struct QUEUE
int ID vector<int> Vehicle_ID int Navteq_ID

Abbildung 28: Struktur Fahrzeugschlange

Für die Realisierung werden zu Beginn für jeden Straßenabschnitt alle darin vorkommenden Fahrzeuge mit Nachbarfahrzeug ermittelt und gesondert zwischengespeichert. Im späteren Verlauf werden dann durch das Abgreifen der Nachbarschaft auch die Fahrzeuge ohne Nachbarschaft mit in der Schlangengenerierung berücksichtigt. Jedes Fahrzeug besitzt pro NAVTEQ ID nach der Reduktion in Kapitel 3.2.5 nur noch ein Nachbarfahrzeug. Der Programmablauf nachfolgend dargestellt.

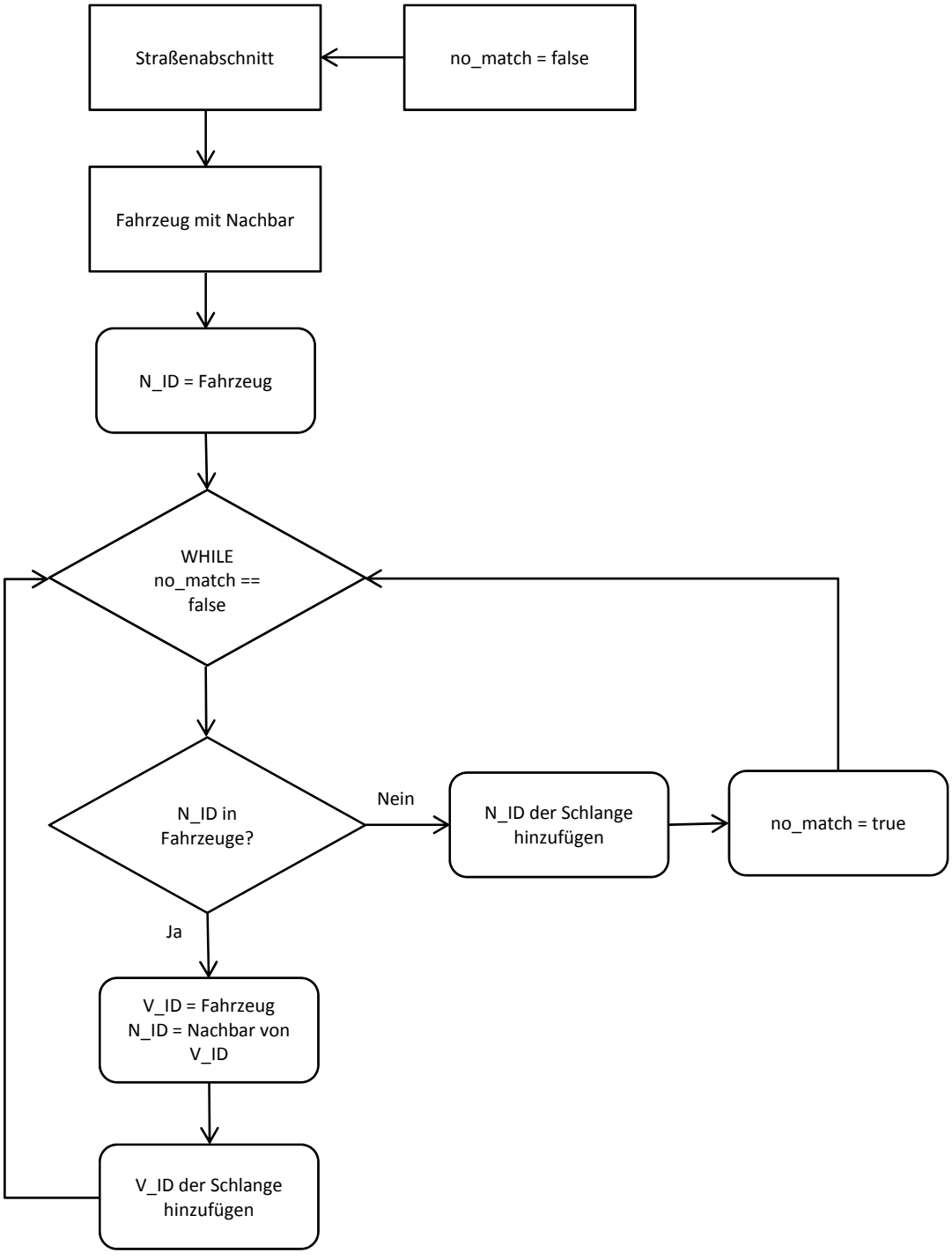


Abbildung 29: Programmablauf - Erfassung der Fahrzeugschlangen

Die Vorgehensweise bezieht sich hier immer auf einen Straßenabschnitt. Sukzessiv werden die vorab zwischengespeicherten Fahrzeuge mit ihrem Nachbarfahrzeug abgegriffen. Nach dem vorherigen Programmabschnitt ist pro Straßenabschnitt nur noch ein Nachbarfahrzeug vorhanden. Für jedes Fahrzeug wird ein leerer Vektor deklariert, diesem später die Fahrzeugschlange hinzugefügt wird. Daneben wird die Variable „no_match“ mit dem Datentyp Bool deklariert und mit dem Wert false initialisiert. Das erste Fahrzeug erhält den Variablennamen N_ID.

Der Ablauf innerhalb der While-Schleife erfolgt solange, bis die Variable „no_match“ den Wert „true“ annimmt. Innerhalb der Schleife wird unter den zwischengespeicherten Fahrzeugen nach dem Nachbarfahrzeug „N_ID“ gesucht, um dann Zugang zu dessen Nachbarfahrzeug zu erhalten. Wurde „N_ID“ unter den Fahrzeugen gefunden, erhält dieses Fahrzeug den Namen „V_ID“ und das zugehörige Nachbarfahrzeug wird zu „N_ID“, nach dessen Fahrzeug ID im nächsten Schleifendurchlauf von Neuem gesucht wird. Die Fahrzeug ID von V_ID wird der Fahrzeugschlange hinzugefügt. War die Suche erfolglos, so wird die Fahrzeug ID von „N_ID“ der Fahrzeugschlange hinzugefügt und die Variable „no_match“ erhält den Wert „true“. Nach Erhalt dieses Wertes wird die While-Schleife abgebrochen.

Dieser Vorgang wird für alle Fahrzeuge durchgeführt. Es kommt zu Problemen, wenn Fahrzeuge sich gegenseitig als Nachbarn besitzen. Die Suche nach dem Fahrzeug würde somit immer erfolgreich sein, was zu einer Endlosschleife führt. Um dieses Problem zu entgehen, wird eine zusätzliche Variable mit dem Datentyp Bool deklariert. Nach jedem Fahrzeugfund wird geprüft, ob das gefundene Fahrzeug das letzte Element der Fahrzeugschlange ist. Ist dies der Fall wird ebenso die While-Schleife abgebrochen. Am Ende existiert für jedes Fahrzeug eine Fahrzeugschlange, in der es an erster Position steht. Somit existiert eine Vielzahl von Schlangen, aus denen die Originalschlange identifiziert werden muss.

In Abbildung 30 ist solch eine Situation dargestellt. Für jedes Fahrzeug wurde eine Fahrzeugschlange generiert. Allerdings wird davon nur die Schlange benötigt, die alle Fahrzeuge beinhaltet. Daher muss aus dem Erzeugnis die Originalfahrzeugschlangen identifiziert werden.

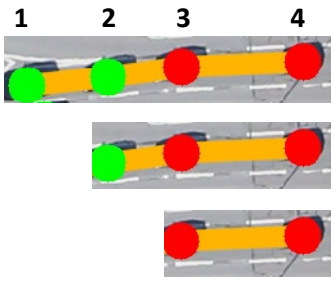


Abbildung 30: Teilschlangen

In der Abbildung umfasst die erste Schlange alle Fahrzeuge und stellt somit die zu identifizierende dar. Alle anderen Fahrzeugschlangen sind unvollständig und werden für den weiteren Programmverlauf nicht benötigt. Um diese Originalfahrzeugschlange zu ermitteln wird für jedes Fahrzeug geprüft, ob es im Bestand anderer Fahrzeugschlangen ist. Ist dies der Fall, bleibt lediglich die Fahrzeugschlange bestehen, die die meisten Fahrzeuge umfasst. Demzufolge wird jedes Fahrzeug nur einer Fahrzeugschlange zugewiesen und die Teilschlangen werden als Folge eliminiert.

3.3. Identifikation von Straßenkreuzungen und Ampeln

Für die anstehende Unterscheidung ist ein Bezug zu den Straßenkreuzungen und Ampeln von Vorteil. Fahrzeuge, die sich in unmittelbarer Nähe von Straßenkreuzungen oder Ampeln befinden nehmen mit einer höheren Wahrscheinlichkeit am Verkehr teil. Für die Ampelpositionen liegt ein Datensatz von München vor. Die Straßenkreuzungen müssen allerdings im Vorfeld lokalisiert werden.

3.3.1. Template Matching

Die Straßenkreuzungen werden auf Basis der NAVTEQ Daten ermittelt. Dies wird mithilfe des Verfahrens Template Matching realisiert. Gesucht werden allerdings nur die Kreuzungen der stark befahrenen Straßenabschnitte. Aus diesem Grund wird auf Basis der NAVTEQ Daten ein Binärbild erzeugt, wobei nur Straßenabschnitte mit der Straßenkategorie eins und zwei mit einbezogen werden (siehe Abbildung 31). Als Linienbreite wurden 10 Pixel gewählt.

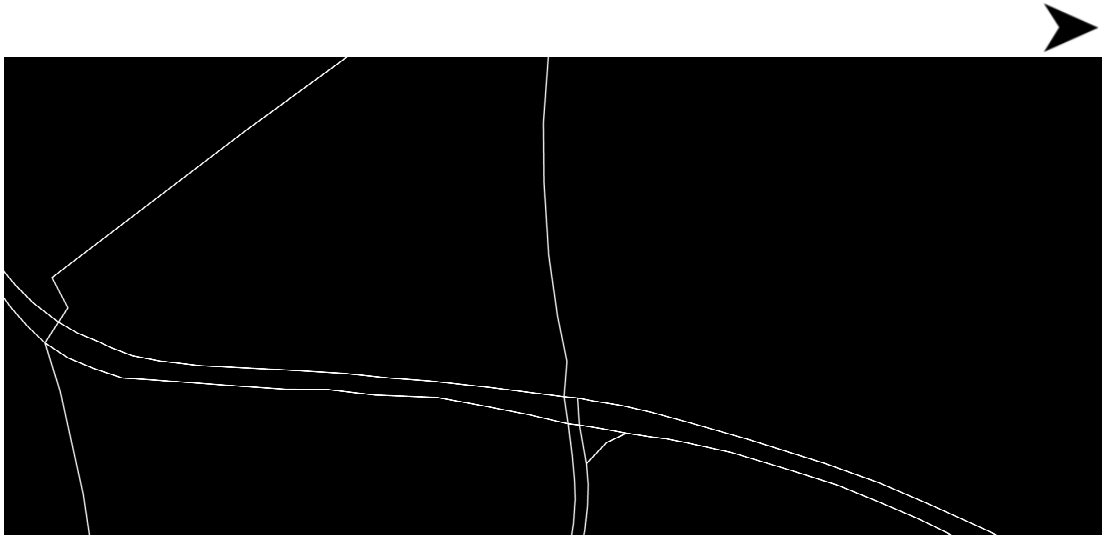


Abbildung 31: Binärbild NAVTEQ

An jeder Stelle, wo die Linien im Bild ein Kreuz bilden, befindet sich eine gesuchte Kreuzung. Diese Stellen im Bild werden mithilfe des Verfahrens Template Matching lokalisiert. Es werden dabei alle Stellen im Ausgangsbild ermittelt, die mit dem Template eine hinreichende Ähnlichkeit aufweisen (Demant et. al., 2011, S. 109). Für diesen Vorgang wird die OpenCV Bibliothek eingesetzt. Zu Beginn wird das Ausgangsbild definiert, in dem nach dem Template gesucht wird. Dies ist in diesem Fall das Binärbild der Straßenabschnitte. Daneben wird für das Template eine Bildmatrix erzeugt, die ein Kreuz darstellt (siehe Abbildung 32). Die Linienbreite beträgt hier ebenso 10 Pixel.

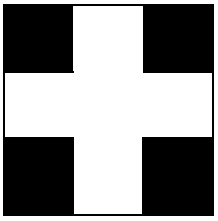


Abbildung 32: Template

Dieses Template wird an jeder Stelle des Ausgangsbildes platziert und ein Ähnlichkeitsmaß berechnet. Hierfür bietet die Programmbibliothek mehrere Vorgehensweisen an (OpenCV - Template Matching, 2017). In diesem Anwendungsfall wurde die Grauwertkorrelation als Verfahren verwendet. Die Berechnung wird für jeden Bildpunkt durchgeführt (Demant et. al., 2011, S. 110).

$$R(x,y) = \sum_{x',y'}\left(\mathrm{T}(x',y') * \mathrm{I}(x+x',y+y')\right)$$

(OpenCV - Template Matching, 2017)

Das Egebnis der Berechnung ist eine Bildmatrix, wobei hier der Pixelwert jeweils das berechnete Ähnlichkeitsmaß entspricht. Hiermit kann für jede Stelle im Bild das Ergebnis der Grauwertkorrelation abgelesen werden. In dieser Arbeit wurden alle Stellen im Bild als Straßenkreuzung herangezogen, deren Pixelwert und somit das Ähnlichkeitsmaß mindestens 0,75 betragen. Dieser Wert wurde so gewählt, dass Kreuzungen auch mit drei Ausfahrten erkannt werden.

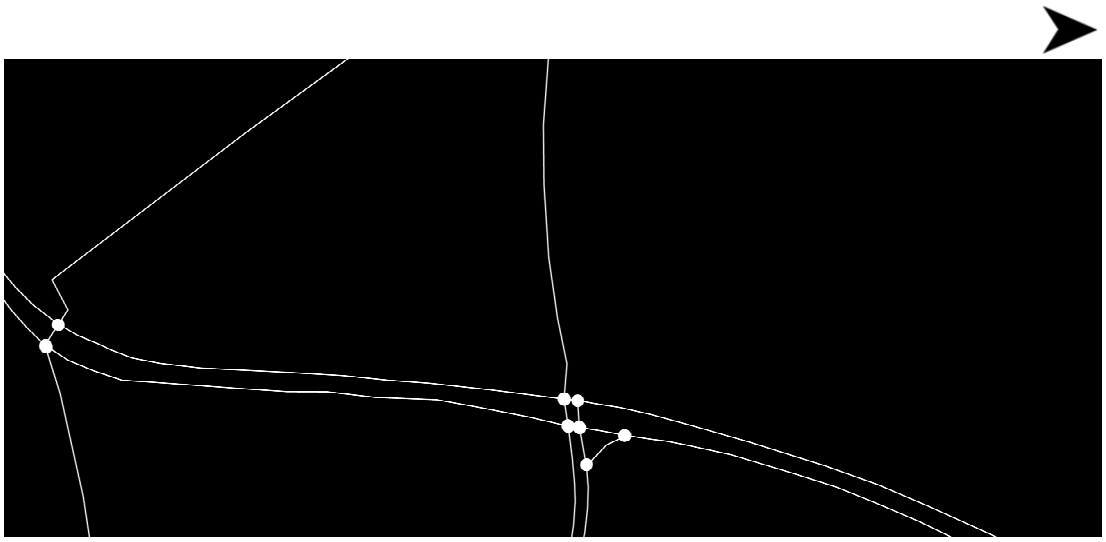


Abbildung 33: Ergebnis Template Matching

In Abbildung 33 sind die gefundenen Straßenkreuzungen mit einem weißen Kreis markiert. Die Straßenkreuzungen werden anschließend mithilfe der Struktur „Crossroad“ in einem Vektor abgespeichert.

struct CROSSROAD
int x int y

Abbildung 34: Struktur Straßenkreuzung

3.3.2. Prüfung auf Fahrspurkategorie

Zusätzlich zu den Koordinaten werden den Straßenkreuzungen und Ampeln die Fahrspurkategorie als Eigenschaft hinzugefügt. Die Fahrspurkategorie gibt Auskunft über die Anzahl der Fahrbahnen auf einem Straßenabschnitt. Sie beeinflusst die Größe vom Umkreis, in dem nach den Fahrzeugen im weiteren Verlauf gesucht wird.

struct CROSSROAD
int x
int y
int LaneCategory

Abbildung 35: Struktur Straßenkreuzung – LaneCateogry

struct TRAFFICLIGHT
int id
int x
int y
double E
double N
int LaneCategory

Abbildung 36: Struktur Ampel - LaneCategory

In Abbildung 37 sind die Straßenabschnitte mit der Fahrspurkategorie zwei dargestellt. Diese Information steht in den Eigenschaften jedes NAVTEQ Stützpunktes.

Abbildung 37: NAVTEQ Linien mit Fahrspurkategorie zwei

Um den Objekstrukturen diese Information übergeben zu können, muss für jede Straßenkreuzungs- und Ampelposition ermittelt werden, ob diese sich auf einem Straßenabschnitt mit der Fahrspurkategorie zwei befinden. Hierfür wird erneut ein Binärbild von den NAVTEQ Linien erzeugt, wobei hier die Fahrspurkategorie zwei betragen muss. Somit werden nur Straßenabschnitte ins Bild gezeichnet, die zwei Fahrbahnen besitzen. Die Straßenabschnitte werden mit dem Pixelwert 255 repräsentiert. Als Linienbreite wurden 22m gewählt.

Für jede Straßenkreuzung wird anschließend im Bild der Pixelwert abgefragt. Falls dieser 255 entspricht, befindet sich die Straßenkreuzung auf einem Straßenabschnitt, der zwei Fahrbahnen besitzt. Diese Abfrage wird ebenso für die Ampelpositionen durchgeführt. Dieses Ergebnis wird jeweils in der Objektstruktur berücksichtigt.

3.3.3. Umfeld von Straßenkreuzungen und Ampeln

Nachdem die Positionen der Ampeln und Straßenkreuzungen bekannt sind, wird für jedes Fahrzeug geprüft, ob es sich in der Nähe einer Ampel oder Straßenkreuzung befindet. Diese Information wird in der Objektstruktur berücksichtigt (siehe Abbildung 38).

struct VEHICLE
int ID
int tracked
int x
int y
double E
double N
double driving_distance
double speed
double driving_direction
vector<NEIGHBOR> Neighbors
vector<NEIGHBOR> NeighborFromVehicleID
bool Crossroad = false;
bool TrafficLights = false;

Abbildung 38: Struktur Fahrzeug - Kreuzung/Ampel

Hierfür wird der Datentyp Bool verwendet. Für alle Fahrzeuge steht die Straßenkreuzung und Ampel vorweg auf „false“. Falls im späteren Verlauf eine Ampel oder Straßenkreuzung im nahen Umkreis festgestellt wird, wird dieser Wert auf „true“ gesetzt.

Zu diesem Zweck wird ein Binärbild erzeugt, in dem alle Straßenkreuzungen als Kreis mit dem Pixelwert von 255 eingezeichnet werden. Der jeweilige Kreisradius ist abhängig von der Fahrspurkategorie. Straßenkreuzungen und Ampeln auf mindestens zweibahnigen Straßenabschnitten benötigten einen größeren Radius. Für diese wurde daher ein Radius von 36m verwendet. Für die einbahnigen Straßenkreuzungen wurde ein Radius von 17m gewählt.

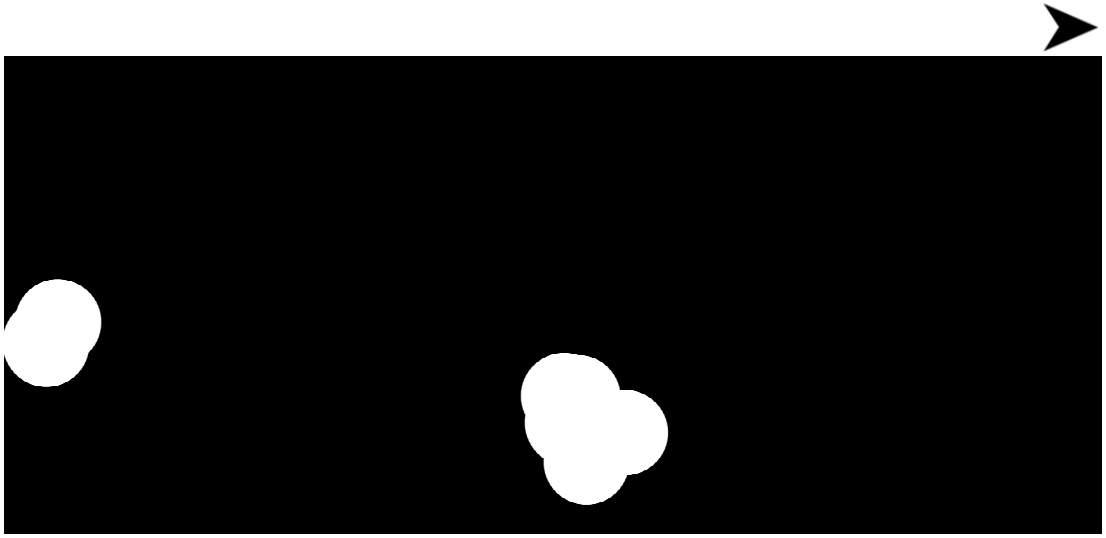


Abbildung 39: Binärbild - Umfeld Straßenkreuzungen

Da es sich bei den Kreuzungspunkten um den Schnittpunkt der Straßenachsen handelt, ist dieser Wert angebracht. Darüber hinaus muss mitberücksichtigt werden, dass viele Haltelinien einige Meter abseits von der Straßenkreuzung sind. Nach der Erzeugung des Binärbildes wird für jede Fahrzeugposition der Pixelwert im Bild abgefragt. Entspricht dieser den Wert 255, so befindet sich das Fahrzeug in der Nähe einer Straßenkreuzung und der Wert des Attributes „Crossroad“ wird auf „true“ gesetzt. In der nachfolgenden Abbildung sind alle Fahrzeuge, die sich im Umfeld von Straßenkreuzungen befinden, rot markiert.

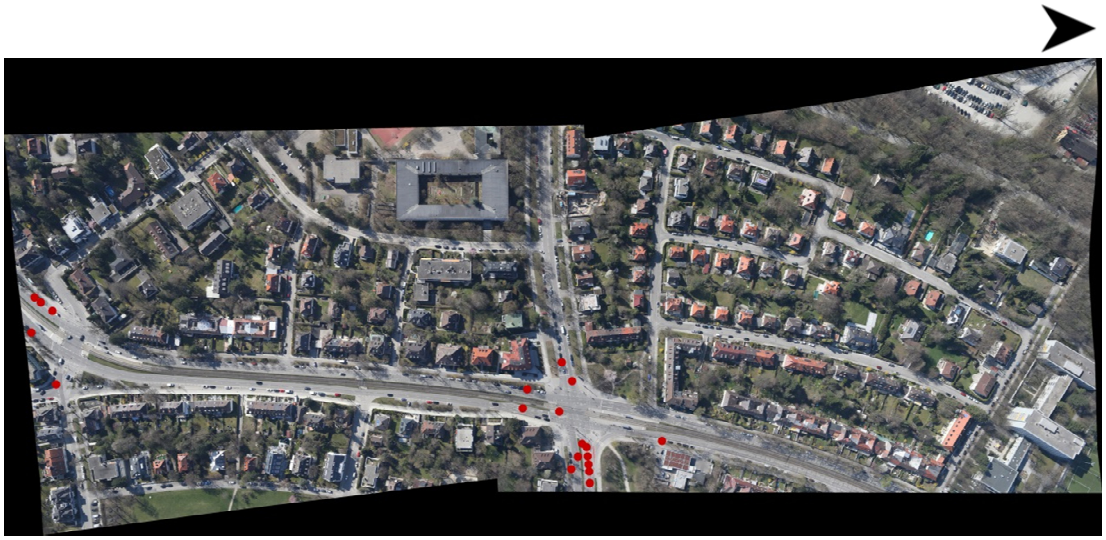


Abbildung 40: Fahrzeuge im Umfeld von Straßenkreuzungen

Bei den Ampeln ist die Vorgehensweise äquivalent. Es wird ein Binärbild mit den Ampelpositionen erzeugt. Diese werden ebenso wie die Straßenkreuzungen als Kreis dargestellt. Der Radius ist ebenso abhängig von der Fahrspurkategorie.

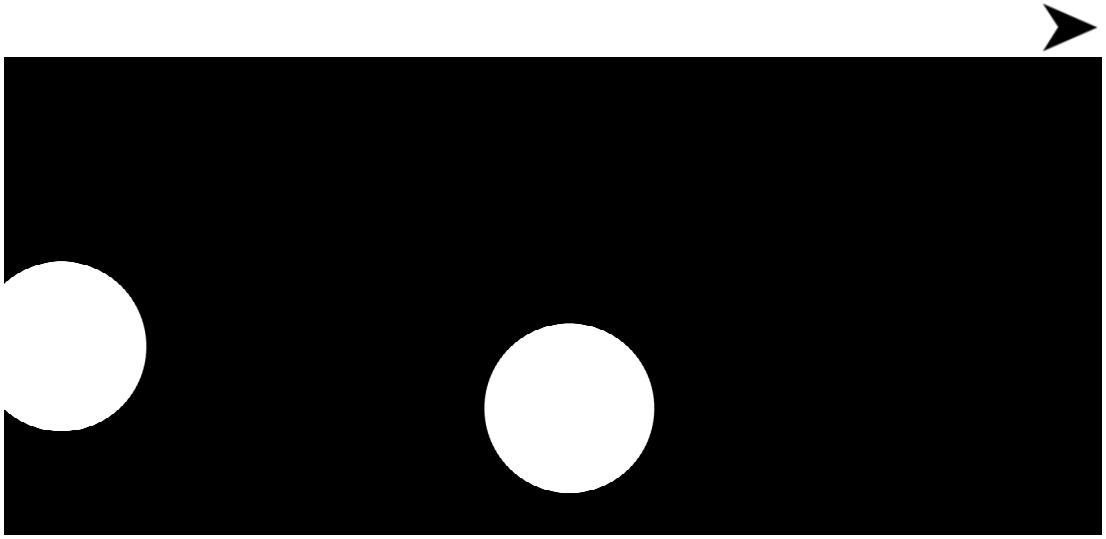


Abbildung 41: Binärbild - Umfeld Ampeln

Da sich bei zweibahnigen Straßenabschnitten die Ampelpositionen von den Positionen der Straßenkreuzungen unterscheiden, wird hier der Radius verdoppelt (siehe Abbildung 41). Bei Straßenkreuzungen auf zweibahnigen Straßenabschnitten existieren vier Kreuzungspunkte, die abseits vom realen Mittelpunkt der Kreuzung liegen. Für die Ampel ist im Gegenzug eine exakte Position vorhanden.

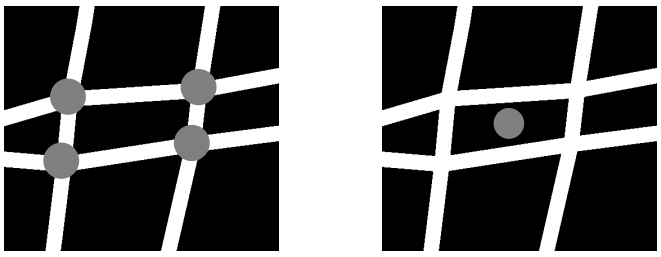


Abbildung 42: Positionen Straßenkreuzung (links) und Ampel (rechts)

Im Anschluss erfolgt die Abfrage der Pixelwerte für die Fahrzeugpositionen im Bild. Wenn der Wert 255 entspricht, befindet sich das Fahrzeug in der Nähe einer Ampel und der Attributwert von

„TrafficLight“ wird auf „true“ gesetzt. In Abbildung 43 sind die Fahrzeuge rot markiert, die sich im Umkreis einer Ampelanlage befinden.

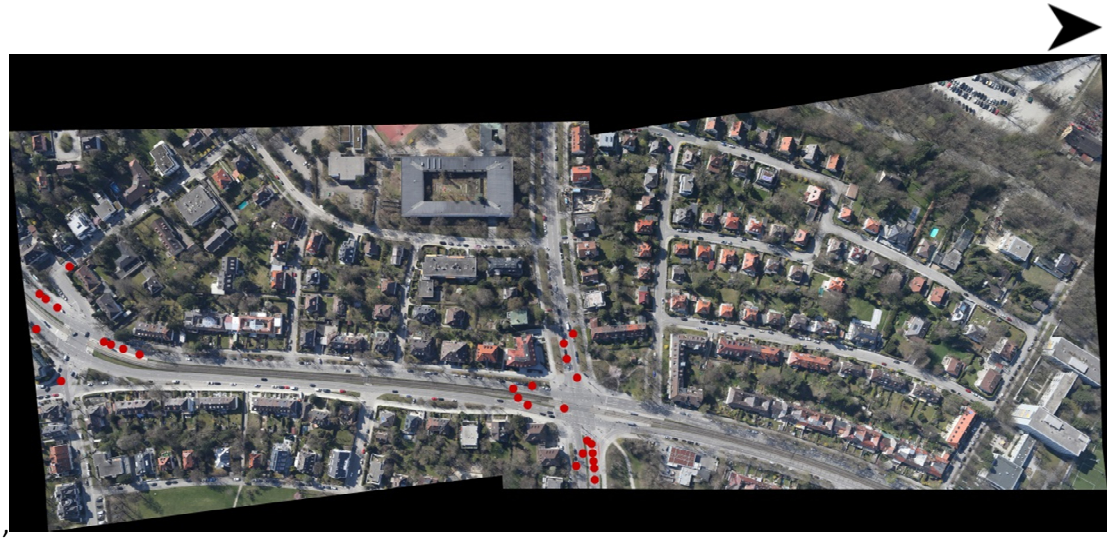


Abbildung 43: Fahrzeuge im Umfeld von Ampeln

3.4. Fuzzy-Logik

Die finale Unterscheidung in am Verkehr teilnehmende und parkende Fahrzeuge basiert auf Fuzzy-Logik. Für die Unterteilung wird ein Fuzzy-System aufgebaut. Auf Basis von vorab definierten Regeln werden hierbei verschiedene Sachverhalte miteinander verknüpft (Bothe, 1995, S.1).

Dies sind in dieser Arbeit die Fahrgeschwindigkeit, Fahrzeugdichte, Straßenkreuzungen, Ampeln, Straßenkategorie und Fahrspurkategorie. Fährt ein Fahrzeug beispielsweise langsam und befindet sich im Umkreis einer Ampel oder Straßenkreuzung, so parkt es mit einer niedrigeren Wahrscheinlichkeit als die Fahrzeuge, die sich nicht im Umkreis befinden. Die Fahrzeugschlangen und die Fahrzeuge ohne Nachbarschaft werden in verschiedenen Fuzzy-Systemen separat betrachtet.

Fuzzy-Logik wurde von L. A. Zadeh im Jahre 1965 entwickelt und ist eine Theorie der unscharfen Mengen (Noll, 2009, S. 25). Eine scharfe Menge A wird immer mithilfe einer Funktion μ_A beschrieben, die nur die Werte 1 und 0 annehmen kann (Noll, 2009, S. 26).

$$\mu_A(x) = \begin{cases} 1, & \text{für } x \in A \\ 0, & \text{sonst} \end{cases}$$

Für die Beschreibung von Eigenschaften, wie beispielsweise die Geschwindigkeit eines Fahrzeuges, ist diese zweiwertige Logik ungeeignet. Es werden fließende Übergänge zwischen Zugehörigkeit und Nichtzugehörigkeit zu einer Menge benötigt (Noll, 2009, S.27). Eine unscharfe Menge wird mithilfe einer Zugehörigkeitsfunktion definiert. Die Werte liegen hierbei zwischen 0 und 1, somit lassen sich auch gleitende Übergänge der Elementzugehörigkeit zu einer Menge beschreiben (Bothe, 1995, S. 6).

In Abbildung 44 ist die „Geschwindigkeit“ als linguistische Variable dargestellt, die durch mehrere linguistische Terme beschrieben wird. Die linguistischen Werte sind in diesem Fall langsam, schnell und werden mithilfe der Zugehörigkeitsfunktion auf eine numerische Wertskala abgebildet (Bothe, 1995, S.9).

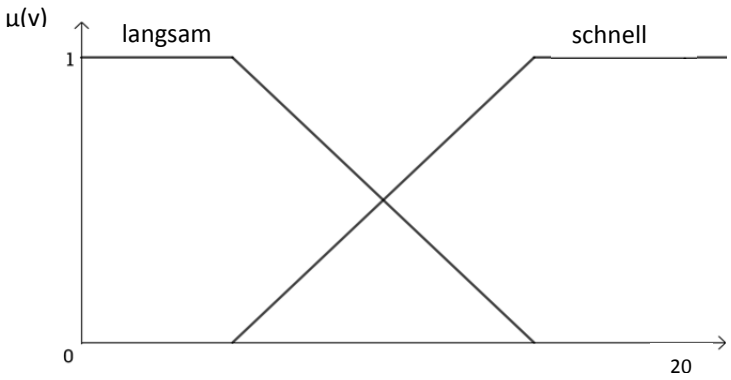


Abbildung 44: Terme der linguistischen Variable Geschwindigkeit

Ein Fuzzy-System erhält einen scharfen Eingangswert und gibt einen scharfen Ausgangswert zurück (siehe Abbildung 45). Mithilfe der Fuzzifikation und Defuzzifikation werden die scharfen bzw. unscharfen Werte in das jeweils andere System überführt, sodass innerhalb des Systems mit unscharfen Werten gearbeitet wird (Bothe, 1995, S.142).

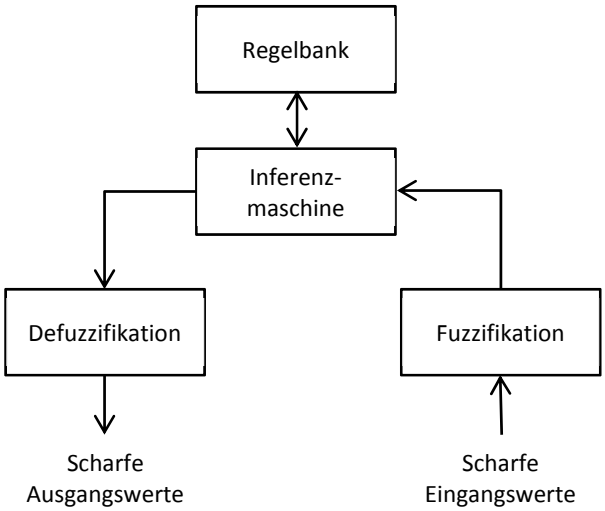


Abbildung 45: Aufbau eines Fuzzy-Systems (Bothe, 1995, S.142)

3.4.1. Fuzzifikation

Bei der Fuzzifikation werden scharfe Werte in unscharfe Werte umgewandelt (Bothe, 1995, S.142). Zu Beginn werden die linguistischen Variablen festgelegt, die anschließend durch Operatoren miteinander verknüpft werden (Bothe, 1995, S.85).

Die Fahrzeugschlangen und Fahrzeuge ohne Nachbarschaft werden innerhalb verschiedener Fuzzy-Systeme separat betrachtet. Aus diesem Grund werden die linguistischen Variablen für beide Sachverhalte im Folgenden getrennt aufgelistet.

A) Linguistische Variablen für Fahrzeugschlangen

   (1) Geschwindigkeit

   (2) Einfluss von Ampel / Kreuzung

   (3) Straßenkategorie

   (4) Verkehrsdichte

B) Linguistische Variablen für Fahrzeuge ohne Nachbarschaft

   (1) Geschwindigkeit

   (2) Einfluss von Ampel / Kreuzung

   (3) Straßenkategorie

Die Terme der linguistischen Variablen werden selbst festgelegt und unter Zuhilfenahme des Endergebnisses iterativ angepasst. Als Terme können Trapez-, Dreiecks- oder Gaußfunktionen mit einem Wertebereich zwischen 0 und 1 verwendet werden (Noll, 2009, S.58). Wichtig ist, dass zwischen den einzelnen Termen eine Überlappung existiert (Bothe, 1993, S. 143).

(1) In Abbildung 46 ist die linguistische Variable „Geschwindigkeit“ mit den dazugehörigen Termen „langsam“ und „schnell“ dargestellt. Die Geschwindigkeit entspricht bei den Fahrzeugschlangen der maximalen Geschwindigkeit aller beteiligten Fahrzeuge. Die Zuordnung einer Variablen zur Fuzzy-Menge erfolgt über die Zugehörigkeitsfunktion.

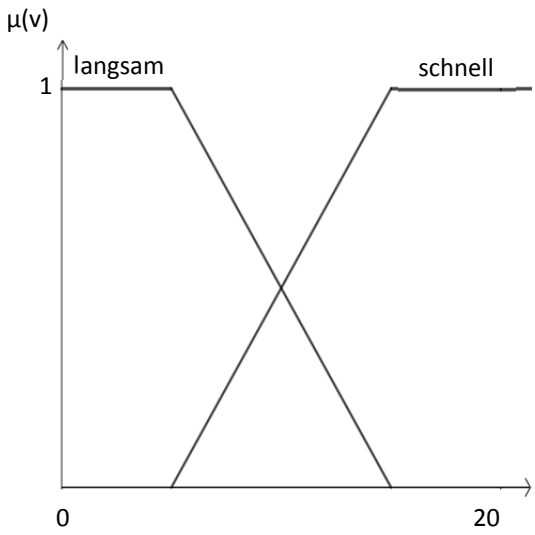


Abbildung 46: Geschwindigkeit

$$\mu_{langsam}(x) = \begin{cases} 1 & x < 5 \\ \frac{15 - x}{10} & 5 \leq x \leq 15 \end{cases}$$

$$\mu_{schnell}(x) = \begin{cases} \frac{x - 5}{10} & 5 \leq x \leq 15 \\ 1 & 15 < x \end{cases}$$

Die Genauigkeit der Fahrgeschwindigkeit wird bei der automatischen Verkehrserfassung von mehreren Faktoren beeinflusst. Dies umfasst die Genauigkeit der Fahrzeugmessung, der Zeitfehler bei der Bildaufnahme und der Maßstabsfehler verursacht vom Digitalen Geländemodell. Unter Berücksichtigung der einzelnen Faktoren ergibt sich eine empirische Standardabweichung kleiner als 5 km/h (Hinz et. al., 2007).

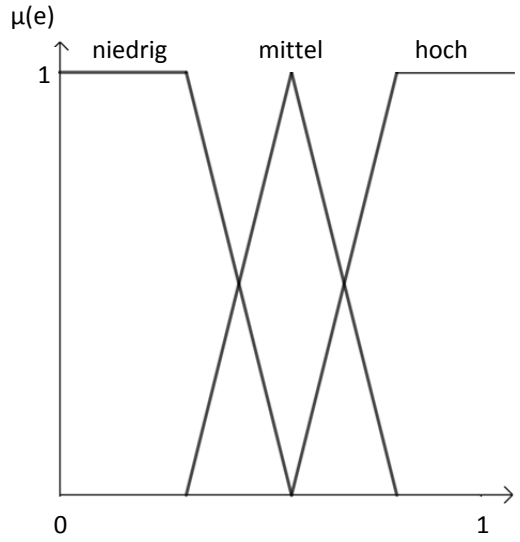
Demnach können alle Fahrzeuge, die eine Geschwindigkeit von weniger als 5 km/h aufweisen, als stehend angesehen werden.

(2) Bei der finalen Unterscheidung ist ebenfalls ein Bezug zu den Straßenkreuzungen und Ampeln erforderlich, da sie Einfluss auf das Fahrverhalten der Verkehrsteilnehmer nehmen. Aus diesem Grund wurde für jedes Fahrzeug in Kapitel 3.3.3. ermittelt, ob in dessen unmittelbarer Umgebung eine Ampel oder Straßenkreuzung existiert. Dieser Sachverhalt wird anschließend mit unterschiedlichen Gewichten im Fuzzy-System berücksichtigt (siehe Tabelle 1).

Situation	**Einflussnahme**
Einspurig / Kreuzung	0,5
Einspurig / Ampel	0,5
Einspurig / Ampel / Kreuzung	0,6
Zweispurig / Kreuzung	0,8
Zweispurig / Ampel	0,8
Zweispurig / Ampel / Kreuzung	0,9

Tabelle 1: Einflussnahme Straßenkreuzungen und Ampeln

Befindet sich eine Ampel und eine Straßenkreuzung im nahen Umkreis des Fahrzeuges, so ist der Einfluss höher als bei lediglich einer vorhandenen Straßenkreuzung. Darüber hinaus können Unterscheidungen hinsichtlich der Fahrspurkategorie vorgenommen werden. Ist die Fahrbahn zweispurig, so ist der Straßenabschnitt stärker befahren und der Einfluss einer Ampel oder Straßenkreuzung ist ebenfalls höher.



$$\mu_{niedrig}(x) = \begin{cases} 1 & x < 0{,}30 \\ \dfrac{0{,}55 - x}{0{,}25} & 0{,}30 \leq x \leq 0{,}55 \end{cases}$$

$$\mu_{mittel}(x) = \begin{cases} \dfrac{x - 0{,}30}{0{,}25} & 0{,}30 \leq x \leq 0{,}55 \\ \dfrac{0{,}80 - x}{0{,}25} & 0{,}55 < x \leq 0{,}80 \end{cases}$$

$$\mu_{hoch}(x) = \begin{cases} \dfrac{x - 0{,}55}{0{,}25} & 0{,}55 < x \leq 0{,}80 \\ 1 & 0{,}80 < x \end{cases}$$

Abbildung 47: Einfluss

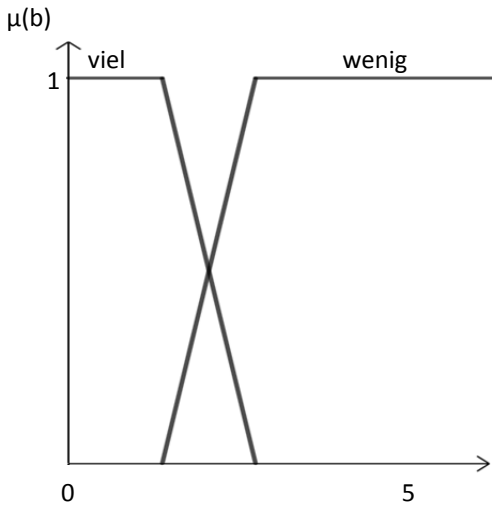
3) Die Straßenkategorie stellt ebenfalls eine linguistische Variable dar. Die Straßenabschnitte wurden hinsichtlich ihrer Funktion und der darauf zulässigen Geschwindigkeit in fünf folgende Straßenkategorien eingruppiert (Digital Map Technology, 2017):

Straßenkategorie	**Charakter**
1	Massenzugangsstraßen mit hoher Fahrgeschwindigkeit
2	Kurze Verbindung innerhalb der Stadt und zwischen Städten
3	Mäßige Fahrgeschwindigkeit innerhalb der Stadt
4	Mäßige Fahrgeschwindigkeit innerhalb von Wohngebieten
5	Geringe Fahrgeschwindigkeit innerhalb von Wohngebieten

Tabelle 2: Straßenkategorie

Diese Informationen sind für die finale Unterscheidung sehr hilfreich, da beispielsweise die Anwohner ihre Fahrzeuge in den Wohngebieten am Straßenrand parken. Mithilfe der Straßenkategorie ist eine spezielle Gewichtung der dort stehenden Fahrzeuge realisierbar.

Für die Definition der Zugehörigkeitsfunktion werden die Straßenkategorien 1-2 in viel befahren und die Kategorien 3-5 in wenig befahren zusammengefasst. Hierfür werden ebenso linguistische Terme definiert (siehe Abbildung 48).



$$\mu_{viel}(x) = \begin{cases} 1 & x < 1{,}50 \\ \dfrac{3{,}00 - x}{1{,}50} & 1{,}50 \leq x \leq 3{,}00 \end{cases}$$

$$\mu_{wenig}(x) = \begin{cases} \dfrac{x - 1{,}50}{1{,}50} & 1{,}50 \leq x \leq 3{,}00 \\ 1 & 3{,}00 < x \end{cases}$$

Abbildung 48: Straßenkategorie

In den nachfolgenden Abbildungen 49 und 50 sind die NAVTEQ Linien mit den Straßenkategorien 1-2 und 3-5 dargestellt.

Abbildung 49: NAVTEQ – Straßenkategorien 1-2

Abbildung 50: NAVTEQ – Straßenkategorien 3-5

(4) Eine weitere linguistische Variable ist die Verkehrsdichte. Diese Variable ist allerdings nur Bestandteil des Fuzzy-Systems der Fahrzeugschlangen. Es wird für jede Schlange die Verkehrsdichte berechnet.

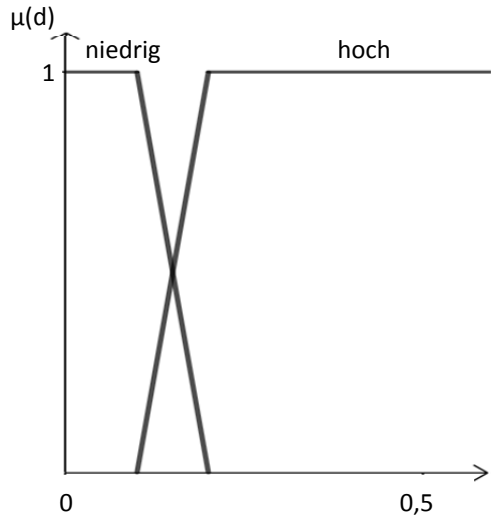
$$D = \frac{n}{\Delta s}$$

D Verkehrsdichte

n Anzahl der Verkehrselemente auf der Fahrzeugschlange

Δs Streckenabschnitt

Der Wert gibt Auskunft darüber, wie dicht die Fahrzeuge hintereinander stehen bzw. fahren. Je größer der Wert, desto kleiner ist der Abstand zwischen den Fahrzeugen. Es kann hierbei davon ausgegangen werden, dass Fahrzeuge bei einem Dichtewert ab 0,15 verkehrsbedingt stehen (Van Aerde & Rakha, 2002).



$$\mu_{niedrig}(x) = \begin{cases} 1 & x < 0{,}10 \\ \dfrac{0{,}20 - x}{0{,}10} & 0{,}10 \leq x \leq 0{,}20 \end{cases}$$

$$\mu_{hoch}(x) = \begin{cases} \dfrac{x - 0{,}10}{0{,}10} & 0{,}10 \leq x \leq 0{,}20 \\ 1 & 0{,}20 < x \end{cases}$$

Abbildung 51: Verkehrsdichte

Nachdem die linguistischen Variablen mit ihren Termen definiert wurden, können für jedes Fahrzeug die Zugehörigkeitswerte ermittelt werden. Zur Veranschaulichung werden nachfolgend zwei Beispiele erläutert.

I. Beispiel – Fahrzeugschlange

Linguistische Variable	**Scharfe Eingangswerte**	**Zugehörigkeitswerte**
Maximale Geschwindigkeit	x = 4	langsam: 1 schnell: 0
Einfluss	x= 0,5	niedrig: 0,2 mittel: 0,8 hoch: 0
Straßenkategorie	x = 2	wenig: 0,33 viel: 0,67
Verkehrsdichte	x = 0,2	niedrig: 0 hoch: 1

Tabelle 3: Beispiel Fahrzeugschlange

Gegeben ist eine Fahrzeugschlange, wobei die maximale Fahrzeuggeschwindigkeit 4 km/h beträgt. Die Fahrzeugschlange befindet sich auf einem einspurigen Straßenabschnitt und in unmittelbarer Nähe einer Straßenkreuzung. Aus diesem Grund wird ein Einfluss von 0,5 vergeben (siehe Tabelle 1). Darüber hinaus stellt der Straßenabschnitt eine kurze Verbindung innerhalb der Stadt dar. Dies entspricht der Straßenkategorie zwei (siehe Tabelle2).

Die Fahrzeuge der Fahrzeugschlange stehen des Weiteren sehr dicht beieinander, sodass die Verkehrsdichte 0,2 beträgt. Mithilfe dieser Daten werden für jede linguistische Variable die Zugehörigkeitswerte zu den jeweiligen Termen berechnet.

II. Beispiel – Fahrzeug ohne Nachbarschaft

Linguistische Variable	**Scharfe Eingangswerte**	**Zugehörigkeitswerte**
Geschwindigkeit	x = 3	langsam: 1 schnell: 0
Einfluss	x = 0	niedrig: 1 mittel: 0 hoch: 0
Straßenkategorie	x = 5	wenig: 1 viel: 0

Tabelle 4: Beispiel – Fahrzeug ohne Nachbarschaft

Bei den Fahrzeugen ohne Nachbarschaft sind die Zugehörigkeitsfunktionen analog. Der einzige Unterschied liegt darin, dass hier die linguistische Variable Verkehrsdichte wegfällt. Darüber hinaus unterscheiden sich die Fuzzy-Regeln.

In diesem Beispiel ist ein Fahrzeug mit einer Geschwindigkeit von 3 km/h gegeben. Das Fahrzeug befindet sich nicht in der Nähe einer Straßenkreuzung oder Ampel und erhält somit einen Einfluss von 0. Darüber hinaus befindet es sich in einem Wohngebiet mit geringer Fahrgeschwindigkeit. Dies stellt die Straßenkategorie 5 dar. Mithilfe dieser Daten und den Zugehörigkeitsfunktionen werden die entsprechenden Zugehörigkeitswerte berechnet.

3.4.2. Fuzzy-Inferenz

Zur Verknüpfung der verschiedenen Sachverhalte werden Fuzzy-Regeln in einer „wenn“-„dann“-Form formuliert (Noll, 2009, S.49). Für die Fahrzeugschlangen und Fahrzeuge, ohne Nachbarschaft werden verschiedene Fuzzy-Regeln aufgestellt.

A) Fahrzeugschlange

R1: **Wenn** Geschwindigkeit = *langsam* **und** Verkehrsdichte = *niedrig* **und** Einfluss = *niedrig* **dann** Parken = *hoch*

R2: **Wenn** Geschwindigkeit = *langsam* **und** Verkehrsdichte = *niedrig* **und** Einfluss = *hoch* **dann** Parken = *niedrig*

R3: **Wenn** Geschwindigkeit = *langsam* **und** Verkehrsdichte = *hoch* **und** Einfluss = *hoch* **dann** Parken = *niedrig*

R4: **Wenn** Geschwindigkeit = *schnell* **dann** Parken = *niedrig*

R5: **Wenn** Geschwindigkeit = *langsam* **und** Verkehrsdichte = *hoch* **und** Einfluss = *niedrig* **dann** Parken = *hoch*

R6: **Wenn** Geschwindigkeit = *langsam* **und** Verkehrsdichte = *niedrig* **und** Einfluss = *mittel* **und** Befahren = *wenig* **dann** Parken = *hoch*

R7: **Wenn** Geschwindigkeit = *langsam* **und** Verkehrsdichte = *niedrig* **und** Einfluss = *mittel* **und** Befahren = *viel* **dann** Parken = *niedrig*

R8: **Wenn** Geschwindigkeit = *langsam* **und** Verkehrsdichte = *hoch* **und** Einfluss = *mittel* **und** Befahren = *viel* **dann** Parken = *klein*

R9: **Wenn** Geschwindigkeit = *langsam* **und** Verkehrsdichte = *hoch* **und** Einfluss = *mittel* **und** Befahren = *wenig* **dann** Parken = *hoch*

B) Fahrzeug ohne Nachbarschaft

R1: **Wenn** Geschwindigkeit = *langsam* **und** Einfluss = *klein* **dann** Parken = *hoch*

R2: **Wenn** Geschwindigkeit = *langsam* **und** Einfluss = *hoch* **dann** Parken = *klein*

R3: **Wenn** Geschwindigkeit = *langsam* **und** Einfluss = *mittel* **und** *Befahren* = *hoch*
dann Parken = *klein*

R4: **Wenn** Geschwindigkeit = *langsam* **und** Einfluss = *mittel* **und** Befahren = *wenig*
dann Parken = *hoch*

R5: **Wenn** Geschwindigkeit = *schnell* **dann** Parken = *klein*

3.4.2.1. Inferenz-Operatoren

Nachdem die Regeln definiert wurden, wird für jede Fahrzeugschlange bzw. Fahrzeug ohne Nachbarschaft ein Vektor des Zugehörigkeitsgrads zur Regelbasis ermittelt. Die Dimension des Vektors entspricht der Anzahl der jeweiligen Regeln. Es werden die Werte für die in der Regel stehenden linguistischen Terme ermittelt und zwischengespeichert. Da es sich bei allen Regeln um eine „und“-Verknüpfung handelt, wird für den Zughörigkeitsgrad zur jeweiligen Regel das Minimum von allen hierfür zwischengespeicherten Werten herangezogen (Noll, 2009, S.64).

Zur Veranschaulichung werden die Beispiele aus Tabelle 3 und 4 dementsprechend weiterverfolgt. Zu Beginn werden die Vektoren „Parken hoch“ und „Parken niedrig“ angelegt, denen die Zugehörigkeitsgrade hinzugefügt werden. Für jede definierte Regel werden die Zugehörigkeitsgrade der dementsprechenden linguistischen Terme ermittelt und zwischengespeichert.
Daraufhin wird das Minimum ermittelt, welches anschließend dem entsprechenden Vektor übergeben wird, auf dem die „Dann“-Aussage der Regel verweist (Mayer et.al., 1993, S. 78).

I. Beispiel – Fahrzeugschlange

Regel	**Minimum**	**„Dann“-Aussage**
R1 = {1; 0; 0,2}	min(R1) = 0	→ Parken hoch
R2 = {1; 0; 0 }	min(R2) = 0	→ Parken niedrig
R3 = {1; 1; 0}	min(R3) = 0	→ Parken niedrig
R4 = {0}	min(R4) = 0	→ Parken niedrig
R5 = {1; 1; 0,2}	min(R5) = 0,2	→ Parken hoch
R6 = {1; 0; 0,8; 0,33}	min(R6) = 0	→ Parken hoch
R7 = {1; 0; 0,8; 0,67}	min(R7) = 0	→ Parken niedrig
R8 = {1; 1; 0,8; 0,67}	min(R8) = 0,67	→ Parken niedrig
R9 = {1; 1; 0,8; 0,33}	min(R9) = 0,33	→ Parken hoch

Tabelle 5: Fuzzy-Inferenz – Beispiel Fahrzeugschlange

I. Beispiel – Fahrzeug ohne Nachbarschaft

Regel	**Minimum**	**„Dann“-Aussage**
R1 = {1; 1}	min(R1) = 1	→ Parken hoch
R2 = {1; 0}	min(R2) = 0	→ Parken niedrig
R3 = {1; 0; 0}	min(R3) = 0	→ Parken niedrig
R4 = {1; 0; 1}	min(R4) = 0	→ Parken hoch
R5 = {0}	min(R5) = 0	→ Parken niedrig

Tabelle 6: Fuzzy-Inferenz – Beispiel Fahrzeug ohne Nachbarschaft

3.4.3. Akkumulation

Der nächste Schritt wird Akkumulation bezeichnet. Die Vektoren „Parken hoch“ und „Parken niedrig“ verfügen über mehrere Fuzzy-Sets. Der Akkumulations-Operator kombiniert die Teilergebnisse mit einer Oder-Verknüpfung zu einem endgültigen Fuzzy-Set. Dies wird mit den t-Conormen realisiert. Da dies eine Oder-Verknüpfung ist, wird der Maximum-Operator verwendet (Mayer et.al., 1993, S. 79).

I. Beispiel – Fahrzeugschlange

Linguistische Variable Parken	**Vektor**	**Maximum**
Parken niedrig	{0; 0; 0; 0; 0,67}	0,67
Parken hoch	{0; 0,2; 0; 0,33}	0,33

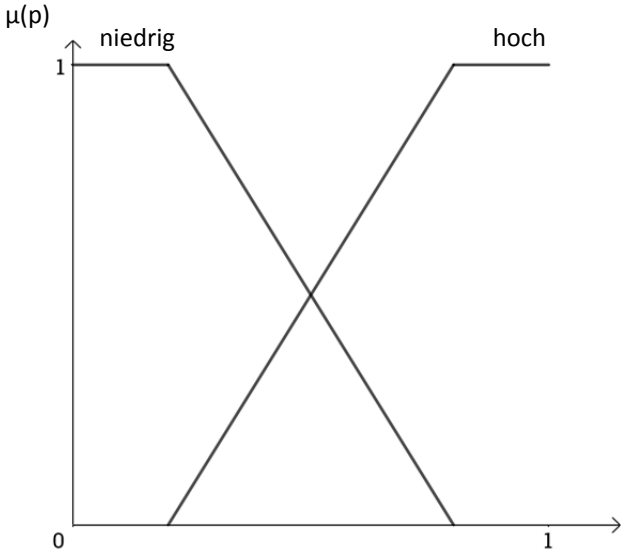
Tabelle 7: Akkumulation – Beispiel Fahrzeugschlange

II. Beispiel – Fahrzeug ohne Nachbarschaft

Linguistische Variable Parken	**Vektor**	**Maximum**
Parken niedrig	{0; 0; 0}	0
Parken hoch	{1; 0}	1

Tabelle 8: Akkumulation – Beispiel Fahrzeug ohne Nachbarschaft

Die Ausgangsvariable ist die linguistische Variable „Parken“, mit den zwei linguistischen Termen „niedrig“ und „hoch“ (siehe Abbildung 52). Das „niedrig“ steht hierbei für „Am Verkehr teilnehmen“ und das „hoch“ für „parkend“.



$$\mu_{niedrig}(x) = \begin{cases} 1 & x \leq 0{,}20 \\ \dfrac{0{,}80 - x}{0{,}60} & 0{,}20 < x \leq 0{,}80 \end{cases}$$

$$\mu_{hoch}(x) = \begin{cases} \dfrac{x - 0{,}20}{0{,}60} & 0{,}20 < x < 0{,}80 \\ 1 & 0{,}80 \leq x \end{cases}$$

Abbildung 52: Linguistische Variable „Parken“

Die Werte für „Parken niedrig“ und „Parken hoch“ stellen die y-Werte dar. Hiermit ergibt sich eine Fuzzy-Menge, mithilfe dieser der scharfe Ausgangswert (x-Wert) ermittelt wird. Diese Berechnung wird in der Defuzzifikation realisiert. In den Abbildungen 53 und 54 sind die Fuzzy-Mengen der zwei Beispiele dargestellt. Um das Endergebnis und somit die finale Unterscheidung in parkende und am Verkehr teilnehmende Fahrzeuge zu erhalten muss aus der grün dargestellten Menge ein scharfer Ausgangswert berechnet werden.

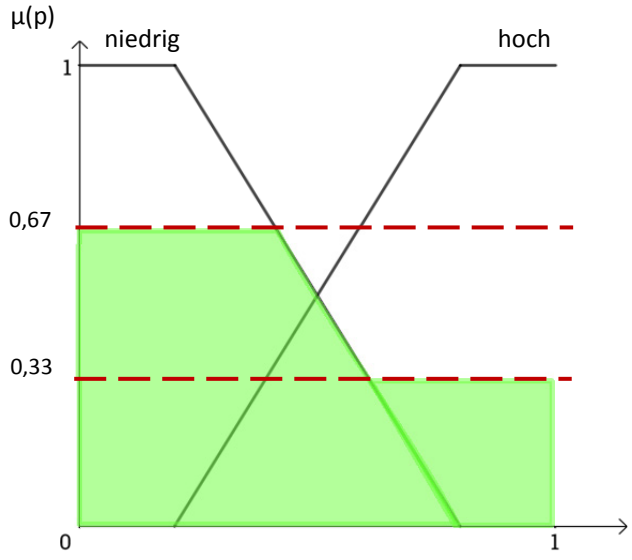


Abbildung 53: Fuzzy-Menge –Beispiel Fahrzeugschlange

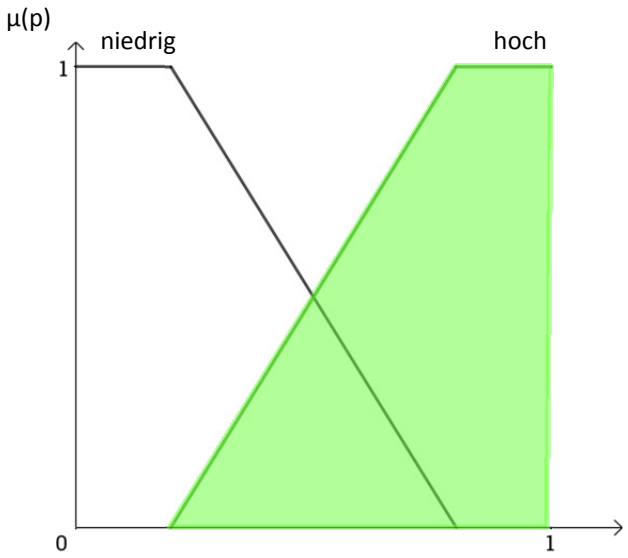


Abbildung 54: Fuzzy-Menge – Beispiel Fahrzeug ohne Nachbarschaft

3.4.4. Defuzzifikation

Für die Berechnung der scharfen Stellwerte existieren mehrere Methoden. In dieser Arbeit wird das Maximum Mittelwertverfahren verwendet. Der scharfe Ausgangswert ergibt sich in diesem Verfahren aus dem arithmetischen Mittel der Maxima (Bothe, 1995, S. 146).

Die Ermittlung erfolgt mit der Zugehörigkeitsfunktion der linguistischen Variablen „Parken“. Mithilfe der zuvor ermittelten y-Werte werden die Stellwerte der lokalen Maxima berechnet. Der scharfe Ausgangswert ist daraufhin das arithmetische Mittel dieser Werte. In den nachfolgenden Abbildungen ist die Defuzzifikation für die Beispiele dargestellt. Die scharfen Werte sind hierbei auf der Abszisse und die dazugehörigen Erfüllungsgrade der linguistischen Terme auf der Ordinate vorzufinden.

I. Beispiel – Fahrzeugschlange

Linguistische Variable Parken	y-Wert	x-Wert	Scharfer Ausgangswert
Parken niedrig	0,67	0,40	0,20
Parken hoch	0,33	0,40	0,70
Maximum der y-Werte		**→ Scharfer Ausgangswert**	
max{0,67; 0.33} = 0.67		→ 0,20	

Tabelle 9: Defuzzifikation – Beispiel Fahrzeugschlange

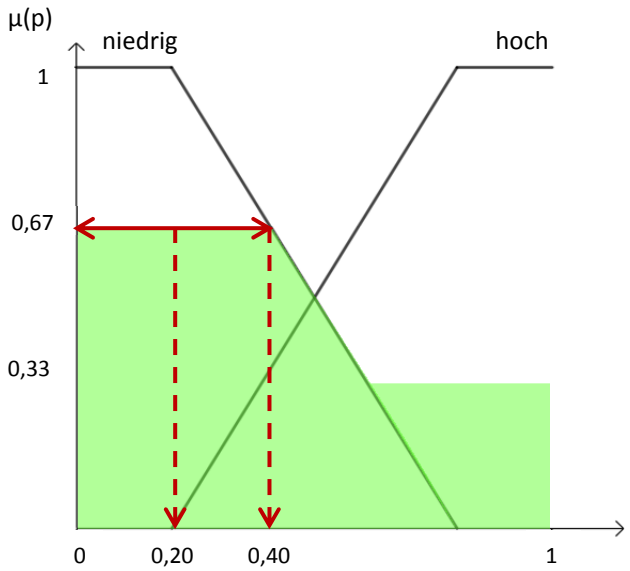


Abbildung 55: Defuzzifikation – Beispiel Fahrzeugschlange

Bei diesem Beispiel beträgt der scharfe Ausgangswert 0,20 mit einem Wahrheitsgehalt von 67%. Daraus lässt sich schließen, dass alle Fahrzeuge der Fahrzeugschlange am Verkehr beteiligt sind.

II. Beispiel – Fahrzeug ohne Nachbarschaft

Linguistische Variable Parken	**y-Wert**	**x-Wert**	**Scharfer Ausgangswert**
Parken niedrig	0,00	0,80	0,40
Parken hoch	1,00	1,00	0,90
Maximum der y-Werte		**→ Scharfer Ausgangswert**	
max{0, 1} = 1,00		→ 0,90	

Tabelle 10: Defuzzifikation – Beispiel Fahrzeug ohne Nachbarschaft

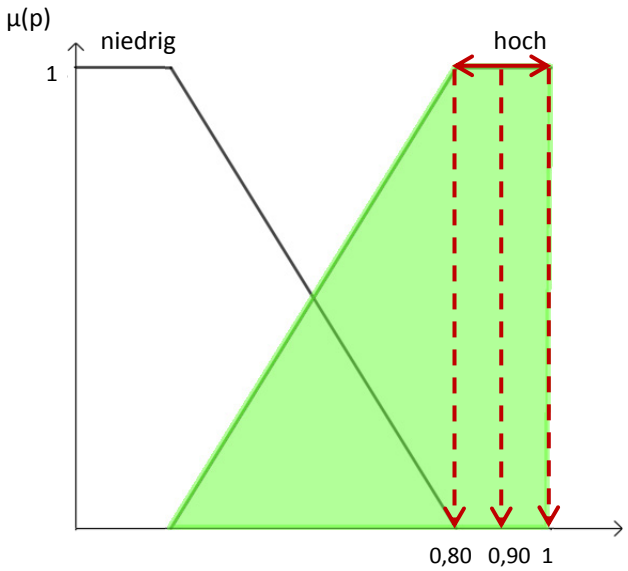


Abbildung 56: Defuzzifikation – Beispiel Fahrzeuge ohne Nachbarschaft

Hier beträgt der scharfe Ausgangswert 0,9 mit einem Wahrheitsgehalt von 100%. Das Fahrzeug parkt demzufolge mit einer hohen Wahrscheinlichkeit.

Zum Abspeichern der Ergebnisse wird die Struktur der Fahrzeugschlange um zwei Attribute ergänzt (siehe Abbildung 57). Dies sind der scharfe Ausgangswert und der Wahrheitsgehalt. Jede Fahrzeugschlange verfügt anschließend über diese beiden Werte in der Objektstruktur.

Struct QUEUE
int ID vector<int> Vehicle_ID int Navteq_ID
float parking_value float validity

Abbildung 57: Struktur Fahrzeugschlange – Fuzzy-Logik

Ebenso wird die Objektstruktur der Fahrzeuge um diese beiden Attribute ergänzt (siehe Abbildung 58). Daraufhin werden jedem Fahrzeug innerhalb einer Fahrzeugschlange die entsprechenden Werte als Eigenschaft übergeben. Den Fahrzeugen ohne Nachbarschaft findet die Übermittlung bereits bei der Defuzzifikation statt. Fahrzeuge, die lediglich detektiert wurden werden von der Fuzzy-Logik ausgeschlossen.

Struct VEHICLE

int ID
int tracked
int x
int y
double E
double N
double driving_distance
double speed
double driving_direction
vector<NEIGHBOR> Neighbors
vector<NEIGHBOR> NeighborFromVehicleID
bool Crossroad = false;
bool TrafficLights = false;

float parking_value float validity

Abbildung 58: Struktur Fahrzeug – Fuzzy-Logik

4. Anwendungsbeispiele

Im Folgenden werden erzielte Ergebnisse vorgestellt. Die Farben der Fahrzeugschlangen entsprechen dem Ergebnis der Fuzzy-Logik. Alle Fahrzeuge innerhalb einer grünen Fahrzeugschlange nehmen am Verkehr teil. Eine rote Schlange beinhaltet im Gegenzug parkende Fahrzeuge. Die punktförmige Darstellung präsentiert die Unterscheidung aller Fahrzeuge.

Abbildung 59: Fahrzeugschlangen - Fuzzy-Logik

Abbildung 60: Fahrzeuge - Fuzzy-Logik

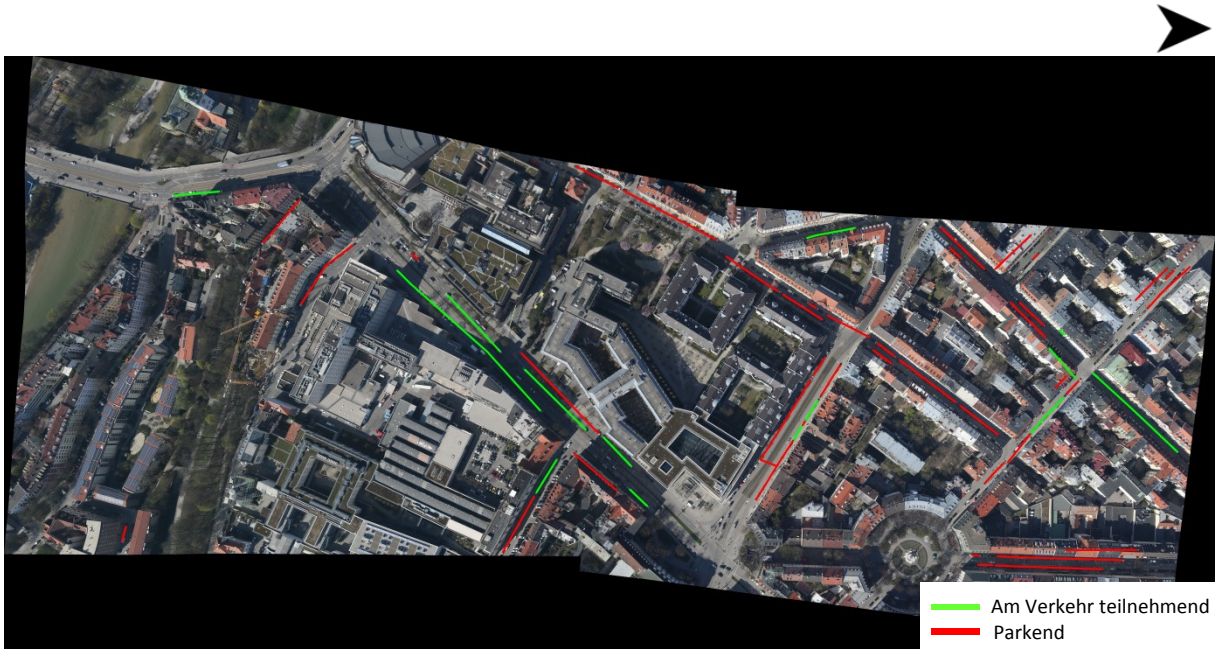


Abbildung 61: Fahrzeugschlangen - Fuzzy-Logik

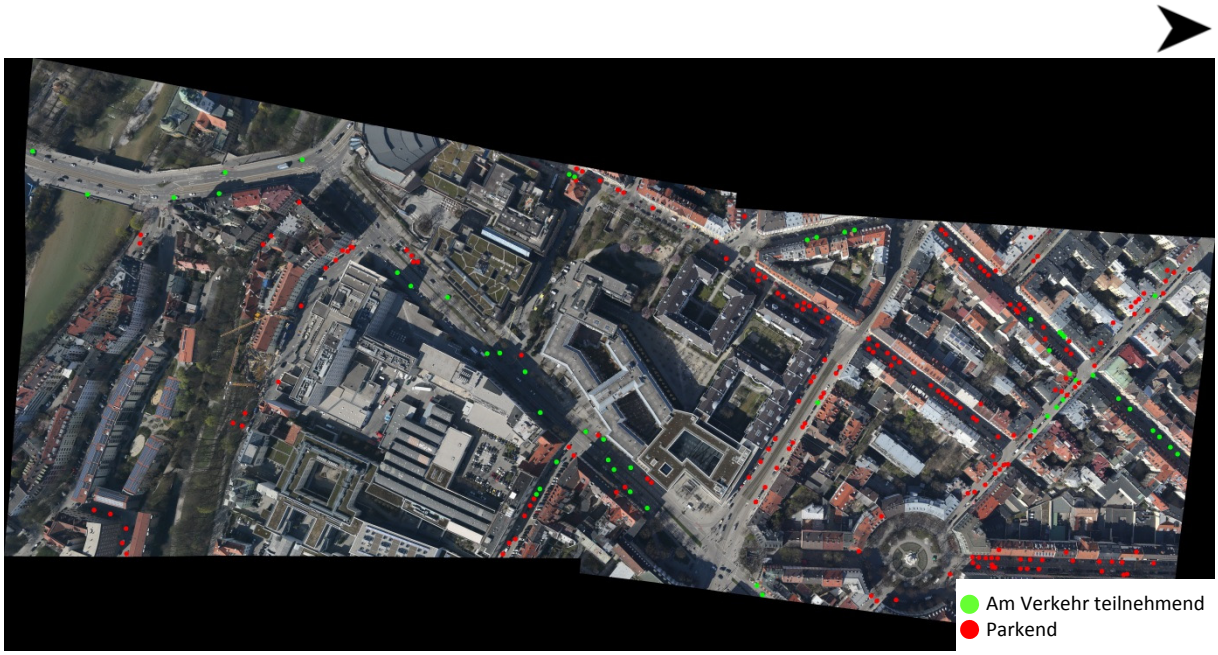


Abbildung 62: Fahrzeuge - Fuzzy-Logik

In Abbildung 64 ist eine Situation vor einer Ampel dargestellt. Die hinteren Fahrzeuge haben hier eine geringe Fahrgeschwindigkeit, die vorderen stehen. Zwei Fahrzeuge auf der mittleren Spur wurden von der automatischen Verkehrserfassung nicht erkannt. Da die Fahrzeuge sich direkt vor einer Ampel befinden und darüber hinaus dicht hintereinander stehen, werden sie mithilfe der Methodik als „Am Verkehr teilnehmen“ markiert. Das Resultat zeigt die nachfolgende Abbildung. Die am Verkehr teilnehmenden Fahrzeuge werden hier als grüne Punkte dargestellt.

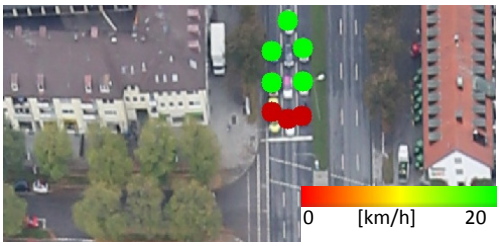


Abbildung 64: Situation vor einer Ampel

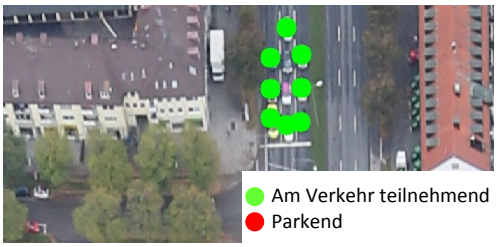


Abbildung 63: Ergebnis - Fuzzy-Logik

Das nachfolgende Beispiel zeigt ein Wohngebiet. Alle Fahrzeuge stehen und der Abstand zwischen den Nachbarfahrzeugen ist groß. Dementsprechend ist die Verkehrsdichte klein. Infolgedessen werden alle Fahrzeuge als „parkend“ markiert.

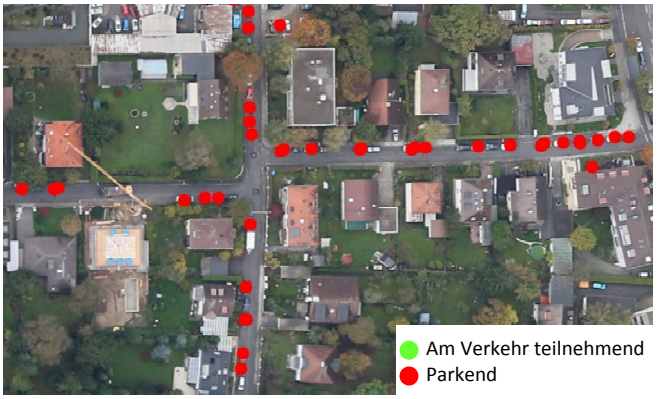


Abbildung 65: Ergebnis – Wohngebiet

5. Fazit und Ausblick

Die in dieser Arbeit entwickelte Methode wird in das operationelle System der Verkehrserfassung aus Luftbildsequenzen übernommen. Für die Beurteilung der erzielten Ergebnisse werden die parkenden und am Verkehr teilnehmenden Fahrzeuge manuell in einem Geoinformationssystem erfasst und anschließend mit dem erzielten Ergebnis der Methodik verglichen. Anhand dessen wird eine Trefferquote von 85-90% ermittelt. Die Verkehrsprozessierung wurde im Postprocessing durchgeführt. Hierbei kann zwischen NAVTEQ und Open Street Map Daten gewählt werden. Bei der Durchführung von Tests war auffällig, dass die Fahrzeugdetektion unter Verwendung von NAVTEQ Daten erfolgreicher war. Aus diesem Grund wurden für diese Arbeit NAVTEQ Daten verwendet. Die entwickelte Methodik unterstützt grundsätzlich beide Datensätze.

Das Ergebnis ist stark von der vorab durchgeführten Verkehrserfassung abhängig. Mögliche Fehldetektionen fließen ebenso in das Endergebnis mit ein. Mit der Bildung von Nachbarschaftsbeziehungen, kann das Endergebnis negativ beeinflusst werden. Existiert in einer Fahrzeugschlange eine Bewegung, werden alle Fahrzeuge innerhalb der Schlange als „am Verkehr teilnehmend“ markiert. Stammt diese Bewegung allerdings von einer Fehldetektion der Verkehrsprozessierung, werden alle Nachbarfahrzeuge vom Algorithmus falsch bewertet (siehe Abbildungen 66 und 67). Bei der Betrachtung der Bilder ist ebenfalls auffallend, dass nicht alle Fahrzeuge vom Algorithmus erkannt wurden. Dadurch werden Informationen vorenthalten, die bei der Bestimmung unter Betrachtung der Nachbarschaftsbeziehungen hilfreich gewesen wären.

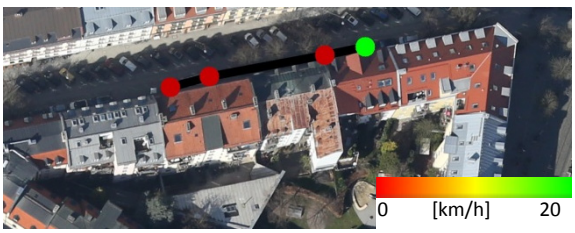


Abbildung 66: Fahrzeugschlangen mit Geschwindigkeit

Abbildung 67: Finale Unterscheidung mit Fuzzy-Logik

Darüber hinaus ist eine hohe Genauigkeit der NAVTEQ Linien erforderlich. Sind diese gegenüber der tatsächlichen Straßenmittelachse versetzt, so sind die Positionen der daraus ermittelten Straßenkreuzungen ebenso fehlerhaft. Des Weiteren ist das Ergebnis der vorab durchgeführten Bündelblockausgleichung ausschlaggebend. Eine exakte Bildgeometrie innerhalb eines Bursts ist für die Bestimmung der Fahrtrichtung und Geschwindigkeit erforderlich.

Eine zusätzliche Herausforderung ist das Mitführen mehrerer NAVTEQ IDs. Da sich mehrere NAVTEQ Linien überlagern, ist diese Vorgehensweise unumgänglich. Schwierigkeiten macht dies bei der Reduktion des Delaunay Graphs auf Fahrzeugschlangen. Ein Fahrzeug darf hierbei nur einen Nachbar besitzen und auch nur einmal als Nachbarfahrzeug fungieren. Dies ist allerdings nur pro Straßenabschnitt realisierbar, da sonst wichtige Nachbarschaftsbeziehungen verloren gehen.

Letztendlich wurde ein gut geeigneter Algorithmus entwickelt, mit dem das bestehende Problem gelöst wurde. Somit sind auch die Verkehrsdaten in der Stadt künftig brauchbar und können neben einer Verkehrslagedarstellung auch künftig zur Verkehrssimulation und Prognosen genutzt werden. Darüber hinaus können mithilfe der parkenden Fahrzeuge indirekt Rückschlüsse auf die Parkräume entlang der Straßen und deren Belegung gezogen werden.

Literaturverzeichnis

Gedruckte Referenz

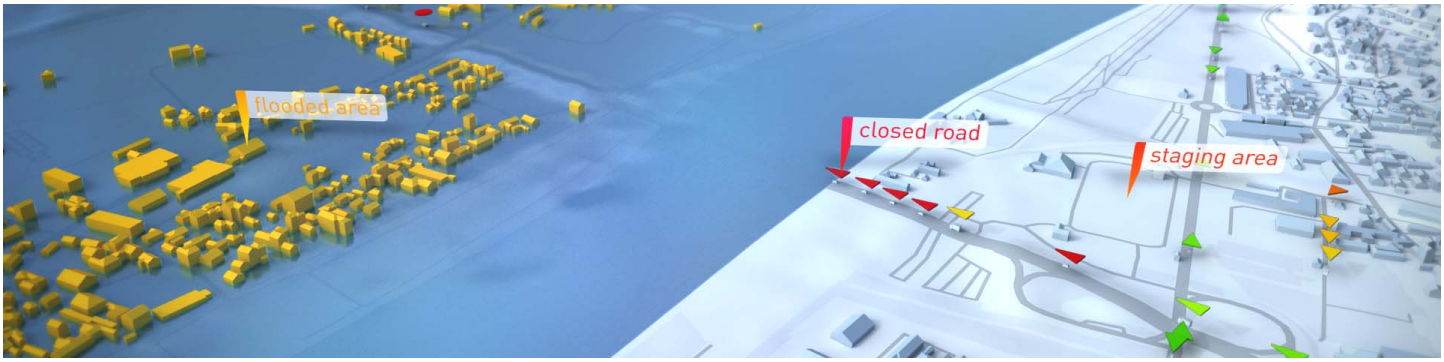
Bothe, 1995	Hans-Heinrich Bothe: Fuzzy Logic Einführung in Theorie und Anwendung, 2. Auflage, Springer-Verlag, Berlin, 1995
Bradski & Kaehler, 2017	Gary Bradski / Adrian Kaehler: Learning OpenCV 3, 1. Auflage, O´Reilley Media, Sebastopol, 2017
Demant et. al., 2011	Christian Demant / Bernd Streicher-Abel /Axel Springhoff: Industrielle Bildverarbeitung, 3. Auflage, Springer Verlag, Stuttgart, 2011
Gruber & Joeckel, 2014	Franz Josef Gruber / Rainer Joeckel: Formelsammlung für das Vermessungswesen, 17. Auflage, Springer Vieweg, Wiesbaden, 2014
Hinz et. al., 2007	Stefan Hinz / Franz Kurz / Diana Weihing / Steffen Suchandt / Franz Meyer / Richard Bamler: Evaluation of Traffic Monitoring based on Spatio-Temporal Co-Registration of SAR Data and Optical Image Sequences, Photogrammetrie Fernerkundung Geoinformation, Jahrgang 2007 (5), Seiten 309-325, ISSN 1432-8364
Knöttner, 2017	Julia Knöttner: Aufbau einer Netzstruktur mit Fahrzeugen basierend auf einer Verkehrserfassung aus Luftbildern, Projektarbeit, 2017
Kurz, et.al., 2012	Franz Kurz / Sebastian Türmer / Oliver Meynberg / Dominik Rosenbaum / Hartmut Runge / Peter Reinartz / Jens Leitloff: Low-cost optical Camera Systems for real-time Mapping Applications, Photogrammetrie Fernerkundung Geoinformation, Jahrgang 2012 (2), Seiten 159-176, DOI: 10.1127/1432-8364/2012/0109, ISSN 1432-8364.
Leitloff et.al., 2014	Jens Leitloff / Dominik Rosenbaum / Franz Kurz/ Oliver Meynberg / Peter Reinartz: An Operational System for Estimating Road Traffic Information from Aerial Images, 2014, Remote Sensing, 6 (11), Seiten 11315-11341, MDPI AG. DOI: 10.3390/rs61111315, ISSN 2072-4292.
Mayer et. al., 1993	Andreas Mayer / Bernhard Mechler / Andreas Schlindwein / Rainer Wolke: Fuzzy Logic, 1. Auflage, Addison-Wesley, Bonn, 1993
Noll, 2009	Patrick Noll: Statistisches Matching mit Fuzzy Logic, 1. Auflage, Vieweg+Teubner, Marburg, 2009
Van Aerde & Rakha, 2002	Michel Van Aerde / Hesham Rakha: Multivariate Calibration of Single Regime Speed-Flow-Density Relationships, 2002, IEEE Xplore, DOI: 10.1109/VNIS.1995.518858

Elektronische Referenz

Digital Map Technology, 2017	Digital Map Technology Has Arrived, http://www.cicomra.org.ar/cicomra2/expocomm/tutorial%208%20dam-%20navteq.pdf, aufgerufen am 21.11.2017, 11:00
Formelsammlung, 2017	http://www.krieger-online.de/markus/hausarbeiten/formelsammlung_physik1.pdf, aufgerufen am 14.11.2017, 11:00
Luftbildkamerasystem, 2017	Neues Luftbildkamerasystem für den Bevölkerungsschutz, http://verkehrsforschung.dlr.de/de/news/neues-luftbildkamerasystem-fuer-den-bevoelkerungsschutz, aufgerufen am 10.11.2017, 15:30
NAVTEQ, 2017	Was sind NAVTEQ Daten, http://downloads.navteq.com/deutsch/products_data_whatis.htm, aufgerufen am 02.11.2017, 11:00
OpenCV - Template Matching, 2017	Template Matching, https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html, aufgerufen am 14.11.2017, 10:00

Bachelorarbeit im Studiengang Vermessung und Geoinformatik

FH·W·S



Trennung von parkenden und am Verkehr teilnehmenden Fahrzeugen basierend auf einer automatischen Verkehrserfassung aus Luftbildern

WS 2017/2018

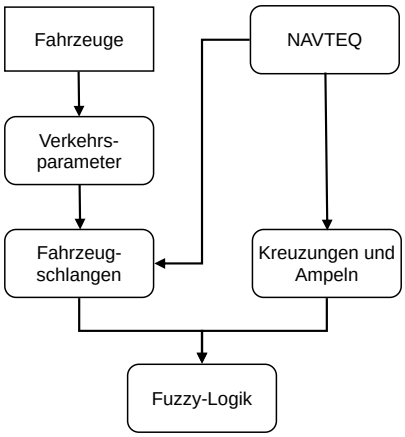
Julia Knöttner; Betreuer: Dr.rer.nat. Dominik Rosenbaum (DLR, IMF-PBA), Prof. Dr. Ing. Ansgar Brunn (FHWS)

DLR

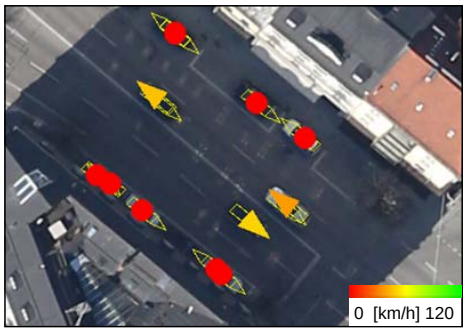
Am DLR werden im Rahmen des Projektes Vabene++ Verkehrsdaten aus Luftbildsequenzen generiert. Mithilfe dieser Daten ist bei einer Katastrophe oder Großveranstaltung eine sofortige Darstellung der aktuellen Verkehrslage herstellbar. Ebenso kann eine Trennung zwischen stehende und fahrende Fahrzeuge realisiert werden. Da allerdings nicht alle stehenden Fahrzeuge ausschließlich parken, werden diese mithilfe dieser Methodik in „parkend“ und „am Verkehr teilnehmend“ unterteilt.

Methodik

- Berechnung der Verkehrsparameter
- Aufbau eines Verkehrsnetzes mittels Delaunay Triangulation
- Ermittlung der NAVTEQ Straßenabschnitte
- Reduktion des Verkehrsnetzes auf Fahrzeugschlangen
- Lokalisation der Straßenkreuzungen mithilfe von Template Matching
- Finale Unterscheidung mit Fuzzy-Logik

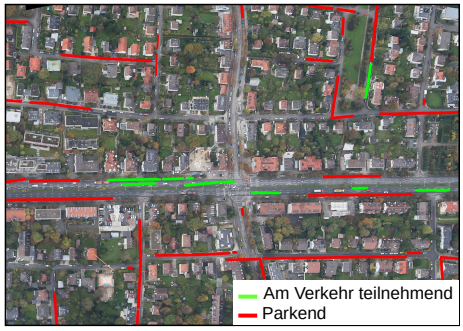


Ergebnisse



Automatische Verkehrserfassung

Kante aus Delaunay Graph



Ergebnis der Fuzzy-Logik